MATHEMATICAL FOUNDATIONS OF DATA-TRAINING-INFERENCE INTERACTION IN LANGUAGE MODELING

Yuchen Li

November 2025
CMU-ML-25-120

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee

Andrej Risteski, Chair
Zico Kolter
Alberto Bietti
Arnaud Doucet
Mikhail Belkin

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

This research was sponsored by: National Science Foundation awards IIS-2211907 and CCF-2238523; an Amazon Research Award; a Google Research Scholar Award; an OpenAI Superalignment Fast Grant.

Keywords: machine learning, theory, language modeling, self-supervised learning, training dynamics, feature learning, generalization, sample complexity, interpretability, mechanistic understanding, sampling, query complexity, inference-time scaling, Transformer, self-attention, masked language modeling, non-autoregressive language models, parallel decoding, verifier-assisted language generation, best-of-N sampling, constrained generation, process verifier, process reward, formal language, Dyck language, context-free grammar, topic model, pseudolikelihood, Markov chain

Abstract

Language models are an amazingly complex technological development—many aspects of which researchers and practitioners currently have very impoverished (mathematical and conceptual) understanding for. This thesis surveys my research aiming to develop a methodology for theoretically reasoning about language models. The methodology is based on mathematically modeling the data generative process by focusing on a few key structures which are commonly present in language data, such as grammars and topics. These structures motivate realistic assumptions on the data, at a level of abstraction which is useful for studying the interaction between data, training, and inference.

Chapter 2 is about training: this chapter covers our works elucidating how Transformer-based models learn simple linguistic structures under common training procedures. Chapter 3 is about inference: this chapter covers our works on inference time scaling when a verifier is available to guide the autoregressive language model generator. Finally, Chapter 4 is about co-designing training and inference procedures, in the context of parallel-efficient language models.

These results connect theoretical analysis of modern neural network architectures to concrete empirical phenomena, and validate our theory in experiments based on synthetic sandboxes and real language data. Through these progresses, my research contributes to developing a mathematical foundation for reasoning about the interactions between data, training, and inference in language modeling, and motivates principled algorithmic design based on understanding and leveraging these interactions.

Acknowledgment

I feel fortunate to have the opportunity to meet many mentors and friends during my years as a Ph.D. student. I am thankful to them for providing me with helpful support or valuable feedback.

First and foremost, I would like to thank my advisor Andrej Risteski, whose advisement has fundamentally expanded my research methodology, perspectives, and skills. I particularly appreciate the methodology that Andrej has taught me over the years, which are useful for identifying clear signals in a fast-developing field like machine learning. I would also like to thank my thesis committee members: Zico Kolter, Alberto Bietti, Arnaud Doucet, and Mikhail Belkin, who formed a dream team for supervising my thesis and provided constructive feedback for my research. Their research deeply inspired me during my PhD. To name a few examples of their papers which shaped my perspectives on some research directions related to my thesis research: theory for generative models (Koehler et al., 2023), language model safety (Zou et al., 2023), Transformer theory (Bietti et al., 2023), diffusion language models (Shi et al., 2024), and theory for deep learning optimization (Belkin, 2021). Consequently, it has been an honor for me to discuss my thesis with them and learn about their feedback. I am thankful to them for their inspiration, encouragement, and support.

I am grateful to many mentors with whom I have had the fortune to collaborate during my PhD: Yuanzhi Li, Jordan T. Ash, Cyril Zhang, Akshay Krishnamurthy, Dylan J. Foster, Boris Dadachev, Kishore Papineni, Sanjiv Kumar, Xian Qian, Ankur Moitra, and Mahdi Soltanolkotabi. Their perspectives significantly broadened my horizons as I learned important methodologies and skills from them in research directions that were new to me, and I appreciate their support and mentorship.

It was also a huge pleasure to work with many peer student collaborators: Kaiyue Wen, Bingbin Liu, Alexandre Kirchmeyer, Aashay Mehta, Yilong Qin, Edoardo Botta, Ashwini Pokle, Jinjin Tian, Dhruv Rohatgi, Abhishek Shetty, Donya Saless, and Zehao Dou. Looking back, I feel thankful and fortunate when I realize that we contributed to each other's growth as researchers, by discussing technical details, teaching each other new skills, and sharing diverse perspectives.

Carnegie Mellon University is an amazing place to learn about and contribute to machine learning research. I am grateful to many people at CMU who supported me during my PhD. In particular, I would like to thank:

- My labmates (PhD students and alumni in Andrej's research group): Bingbin Liu, Tanya Marwah, Elan Rosenfeld, Niki Hasrati, Stephen Huan, Anna Wei, and Jingchu Gai for consistently sharing with me insightful knowledge about research directions of their expertise, and providing constructive feedback during my practice talks. The open and supportive environment in the group has tremendously benefited my growth as a student researcher.

- Machine Learning Department staff: Diane Stidle, Suzanne Lyons Muth, Laura D. Winter, Marlee Bandish, Sara Werner, Russ O'Lare, Dorothy Florence Holland-Minkley, Christy Melucci, Amy Protos, Mary Stech, Joshmin E. Ray, Peter Jhon, Daniel Philip Bird, Christine Martik, and Dan Griffin for managing departmental logistics and supporting me regarding departmental requirements or events.

- My office mates: Kelly He, Ellie Haber, Yiding Jiang, Sam Sokota, and Rebecca Yu for always being supportive through the ups and downs and consistently sharing many thoughts, perspectives, feelings, and observations.

- My cohort of PhD students at Machine Learning Department: Sam Sokota, Yiding Jiang, Valerie Chen, Anna Bair, Che-Ping Tsai, Alex Li, Nupoor Gandhi, So Yeon (Tiffany) Min, Keegan Harris, and Shengyuan Hu for building a trusted social circle and supporting each other (e.g. by organizing social events and forming course study groups), especially when we started our PhD at CMU together in August 2020, during the challenging times of the COVID-19 pandemic which caused all of our courses and research activities to be held remotely for a year.

- Faculty instructors of the courses I assisted in teaching: Nihar Shah, Hoda Heidari, and Andrej Risteski for providing helpful feedback about teaching skills.

- My Speaking Skills Exam committee: William Cohen, Yusha Liu, and Keegan Harris for providing constructive feedback about presentation skills.

- Machine Learning Department faculty: Ameet Talwalkar, Virginia Smith, and Kun Zhang for inviting me to attend some series of their reading groups.

- CMU faculty: Geoffrey Gordon, Graham Neubig, Max Simchowitz, Zachary Lipton, Sivaraman Balakrishnan, Barnabas Poczos, Eric Xing, Rayid Ghani, Aran Nayebi, Aditi Raghunathan, Pradeep Ravikumar, Aaditya Ramdas, Christos Faloutsos, and Albert Gu for discussions which at least partially shaped my perspective about some research directions, about career, or about CMU.

- CMU students and recent alumni: Shanda Li, Luyu Gao, Emmy Liu, Yuda Song, Lingjing Kong, Gaurav Ghosal, Zixin Wen, Omar Chehab, Jeremy Cohen, Paul Liang, Yue Wu, Yewen Fan, Xiangchen Song, Martin Q. Ma, Andy Zou, Christina Baek, Stefani Karp, Hao Zhu, Amrith Setlur, Junhong Shen, Amanda Bertsch, Lingxiao Zhao, Andrew Luo, Runtian Zhai, Saurabh Garg, Chirag Gupta, Fahim Tajwar, Sachin Goyal, Jacob Springer, Swaminathan Gurumurthy, Chris Ki, Lili Chen, Kevin Li, Khurram Yamin, Naveen Raman, Zijun Ding, Aakash Lahoti, Shuyan Zhou, Yingshan Chang, Zhengyang Geng, Hanlin Zhang, Yixuan Xu, Ziqian Zhong, Ken Ziyu Liu, Han Guo, Mingkai Deng, Youngseog Chung, Chenghui Zhou, Robin Schmucker, Che-Ping Tsai, Jun-Ting Hsieh, Jeff Xu, Goutham Rajendran, Tom Yan, Tian Ye, Zhili Feng, Oscar Li, Ziyan Wang, Shaojie Bai, Yi Zhou, Mingjie Sun, Shenghao Wu, Yuting Deng, Zhiqiu Lin, Pengkun Liu, Mingkuan Xu, Stephanie Milani, Arundhati Banerjee, Isaac Liao, Dacheng Li, Rulin Shao, Yuxin Xiao, Chenhao Niu, Steven Kolawole, Jared Fernandez, Lucio Dery, Gokul Swamy, Euxhen Hasanaj, Brandon Trabucco, Shantanu Gupta, Yige Hong, Mariya Toneva, Chun Kai Ling, Yao-Hung Hubert Tsai, Ben Eysenbach, Devendra Chaplot, Weichen Wu, Yuchen Xu, Wenbo Cui, Jiayi Weng, Siping Wang, Qiu Jin, and Zichen Wang for discussions which at least partially shaped my perspective about some research directions, about career, or about CMU.

- Other friends at CMU whom I had the pleasure to meet through various occasions, including those in the Machine Learning Department, other departments in computer science, or other fields. They represent the majority of my social circle at CMU. Our interactions defined the non-technical part of my CMU experience and significantly enriched my memory about Pittsburgh. My PhD journey would not have been so enjoyable without them.

Outside of CMU and besides my co-authors during PhD, I am fortunate to have received helpful mentorship and feedback from the larger research community related to the field of machine learning. I am grateful to many people for their support:

- Faculty research mentors during my undergraduate study: Jiawei Han, A.J. Hildebrand, Pramod Viswanath. It was really inspiring to continue discussing with them about research after I have started my PhD, and learning from them about applying their framework of thinking to new research directions and settings.

- Student research mentors and collaborators during my undergraduate study: Hongyu Gong, Yu Shi, Jiaming Shen, Xinwei He, Yunyi Zhang, Zhengzhi Lou, Naijing Zhang, and Qi Zhu. It was a pleasant and inspiring experience to work with them, and I appreciate them for sharing skills and perspectives with me. These research experiences contributed to my decision to pursue a PhD.

- Senior mentors who shared insightful thoughts or constructive feedback during research discussions: Surbhi Goel, Matus Telgarsky, Ramya Vinayak, Daniel Hsu, Pravesh Kothari, Tengyu Ma, Ahmad Beirami, Molei Tao, Vaishnavh Nagarajan, Saman Biookaghazadeh, Yao Zhao, Han Zhao, Simon Shaolei Du, Zonglin Li, Chong You, Nikunj Saunshi, Jiaming Shen, Daliang Li, Srinadh Bhojanapalli, Ziwei Ji, Yi Zhang, Song Mei, and Yuandong Tian.

- Peers with whom I had in-depth research discussions, during various internships, workshops, conferences, or reading groups: Binghui Peng, Jiachen T. Wang, Sadhika Malladi, Depen Morwani, Dong Won Lee, Yichen Li, Shuo Wen, Ezra Edelman, Jingwen Liu, Thuy-Duong "June" Vuong, Xinyan Hu, Yuqing Wang, Kaifeng Lyu, Abhishek Panigrahi, Sanae Lotfi, Zhengzhi Lou, Yubo Cao, Hongru Yang, Haoxiang Wang, Tianqi Chen, Yibo Jiang, Dylan Zhang, Sidi "Steve" Lu, Lang Yin, Yuzheng Hu, Xinyuan Cao, Jingfeng Wu, and Nived Rajaraman.

Last but not least, I am deeply grateful to many people who are outside of my PhD academic circle. I am thankful to my family, especially my parents and my life partner, for their unconditional support. I would also like to thank my teachers, mentors, coaches, and friends who contributed to my growth. Their support and guidance led me to where I am today, and I will carry forward the learning and positive feelings in my future endeavors.

Contents

1 Introduction **11**

2 How Transformers learn simple linguistic structures **14**

2.1 How Transformers learn topic structure: towards a mechanistic understanding . . . 15

2.1.1 Technical setup . . . 15

2.1.2 Overview of results . . . 18

2.1.2.1 Topic structure is encoded in token embeddings . . . 18

2.1.2.2 Topic structure is encoded in self-attention . . . 18

2.1.2.3 Empirical results . . . 20

2.1.3 Topic Structure Can Be Encoded in Token Embeddings . . . 22

2.1.4 Topic Structure Can Be Encoded in Self-Attention . . . 22

2.1.4.1 The two-stage optimization process of self-attention . . . 23

2.1.4.2 Optimal W^V given uniform attention . . . 23

2.1.4.3 Optimal attention weights . . . 24

2.1.5 Experiments . . . 25

2.1.5.1 Results on synthetic LDA-generated data . . . 25

2.1.5.2 Additional results on learned value matrix $\boldsymbol{W}^V$ . . . 26

2.1.5.3 Additional results on learned attention weights . . . 27

2.1.5.4 Results on natural language data . . . 28

2.1.5.5 Loss landscape with respect to attention weights in the non-asymptotic setting 31

2.1.6 Discussion . . . 32

2.1.6.1 The two-stage optimization process . . . 32

2.1.6.2 Do topic-wise behaviors perfectly correlate with co-occurrence counts? . . . 34

2.1.7 Proofs . . . 35

2.1.7.1 Lemma 2.1.1 on the optimal linear transform when freezing uniform attention 35

2.1.7.2 Proof of Theorem 2.1.1: Optimal Token Embedding . . . 39

2.1.7.3 Optimal $\boldsymbol{W}^V$ when freezing uniform attention without regularization . . . 41

2.1.7.4 Proof of Theorem 2.1.2: case when adding L_2 regularization . . . 41

2.1.7.5 Helping lemmas on masking probabilities . . . 43

2.1.7.6 Implication of topic-wise attention assumption on model output . . . 44

2.1.7.7 Proof of Theorem 2.1.3 (optimal attention when freezing $\boldsymbol{W}^V$ to uniform blocks) . . . 45

2.1.7.8 Optimal attention weights (when freezing diagonal $\boldsymbol{W}^V$) . . . 50

2.1.8 Related Works . . . 55

2.1.9 Conclusion . . . 56

2.2 How Transformers learn context-free grammar: qualitatively different optima and pitfalls of myopic interpretability heuristics . . . 57

2.2.1 Technical setup . . . 59

2.2.2 Theory: optimal attention . . . 61

2.2.2.1 Perfect Balance Condition: Ideal Generalization of Unbounded Length . . . 61

2.2.2.2 Approximate Balance Condition For Finite Length Training Data 63
2.2.2.3 Theory: indistinguishability from a single weight matrix 64
2.2.3 Experiments 65
2.2.3.1 Training Details 66
2.2.3.2 Different Attention Patterns Can Be Learned To Generate Dyck 66
2.2.3.3 Guiding The Transformer To Learn Balanced Attention 67
2.2.3.4 Results on Dyck prefixes 68
2.2.3.5 Additional variants of Dyck languages and architectures 70
2.2.4 Discussion: are interpretable attention patterns useful? 71
2.2.4.1 Can interpretable attention patterns be misleading? 71
2.2.4.2 Are attention heads with interpretable patterns more important? 75
2.2.5 Omitted Proofs in Section 2.2.2 75
2.2.5.1 Proof of Theorem 2.2.1 75
2.2.5.2 Implication of our results to larger models 79
2.2.5.3 Proof of Corollary 2.2.1 79
2.2.5.4 Proof of Theorem 2.2.2 80
2.2.5.5 Proof of Theorem 2.2.3 83
2.2.5.6 Helper lemmas for Theorem 2.2.3 85
2.2.5.7 Additional lemmas 92
2.2.5.8 Discussion on architecture choices 94
2.2.6 Related Work 95
2.2.7 Conclusion 96

3 Improving language model inference scaling through verifier-guided backtracking 98
3.1 Theoretical framework: query complexity of verifier-assisted language generation 99
3.1.1 Technical setup 99
3.1.2 Constrained generation is hard without a verifier 101
3.1.3 Constrained generation with process verifier gets easier 102
3.2 Heuristic algorithm: tokenwise rejection sampling with backtracking 103
3.2.1 Language models trained on synthetic data 104
3.2.1.1 Dyck grammar as a sandbox 104
3.2.1.2 Training the verifier 105
3.2.1.3 Backtracking effectively reduces errors 105
3.2.1.4 Verifier effectively reduces errors 105
3.2.1.5 Tokenwise rejection sampling with backtracking reduces completion errors on unseen OOD prefixes 105
3.2.1.6 Visualizing the language model representations of correct vs. incorrect sequences 106
3.2.1.7 The predicted backtracks were necessary 108
3.2.1.8 Error analysis on the remaining mistakes 109
3.2.1.9 Tokenwise rejection sampling with backtracking maintains diversity 110
3.2.2 Generating test cases with pretrained CodeLlama 111
3.2.2.1 Task setup 111
3.2.2.2 Tokenwise rejection sampling with backtracking improves accuracy 111
3.2.2.3 Tokenwise rejection sampling with backtracking is query efficient 112
3.2.2.4 Examples of prompts and model completions 113
3.2.2.5 Baselines 115
3.2.2.6 Training the verifier 116
3.2.2.7 Full results of CodeLlama experiments in Section 3.2.2 119
3.2.2.8 Visualizing the query efficiency of Tokenwise rejection sampling with backtracking 121

3.2.2.9 Tokenwise rejection sampling with backtracking generalizes better to out-of-distribution prompts . . . 123
3.2.3 Additional ablation experiments on the Tokenwise rejection sampling with backtracking algorithm (Algorithm 1) . . . 126
3.3 Theoretically provable algorithm: value-guided sampling with stochastic backtracking . . . 127
3.3.1 VGB: the Value-Guided Sampling with Stochastic Backtracking algorithm . . . 127
3.4 Discussions . . . 129
3.4.1 Is query efficiency a reasonable notion of efficiency? . . . 129
3.5 Proof intuitions . . . 130
3.5.1 On the hardness of the knapsack problem . . . 130
3.5.2 Proof intuition for Theorem 3.1.1 . . . 130
3.6 Related work . . . 130
3.7 Conclusion . . . 131

4 Co-designing inference and training procedures for parallel-efficient language models 133
4.1 Theoretical framework . . . 135
4.1.1 Setup and notation . . . 135
4.1.2 Asymptotic sample efficiency via functional inequalities . . . 136
4.1.2.1 Masking more is (statistically) better . . . 137
4.1.2.2 Statistical efficiency bounds via mixing time bounds . . . 137
4.1.2.3 Adaptive masking: masked positions depend on the sequence . . . 138
4.1.3 Finite sample bounds and distributional distance . . . 139
4.1.4 Inference-time limitations due to parallelism . . . 140
4.1.4.1 Which Markov Chains are implementable via parallel decoding? . . . 141
4.1.4.2 Markov Chains with dependent transitions can be (much) faster . . . 142
4.2 Experiments . . . 144
4.2.1 Synthetic experiments on Ising model . . . 144
4.2.1.1 Masking more is statistically better for learning synthetic Ising models . . . 144
4.2.1.2 Markov Chains with dependent transitions can be (much) faster in sampling Ising models . . . 146
4.2.2 Parallel Decoding by Iterative Refinement (PaDIR) on real language data . . . 147
4.2.2.1 Inference approach for PaDIR . . . 147
4.2.2.2 Training approach for PaDIR . . . 147
4.2.3 Dataset and evaluation . . . 148
4.2.4 Connecting to theory: quantifying dependency via attention scores . . . 150
4.3 Proofs . . . 152
4.3.1 Proof of Lemma 4.1.2: Generalized information matrix equality . . . 152
4.3.2 Proof of Theorem 4.1.1: Masking more is (statistically) better . . . 154
4.3.3 Generalizations for adaptive masking . . . 156
4.3.3.1 Conditioning on K . . . 156
4.3.3.2 Information matrix equality for adaptive masking . . . 157
4.3.3.3 Proof of Proposition 4.1.1: Dirichlet form for adaptive block dynamics . . . 158
4.3.3.4 Proof of Theorem 4.1.3: Asymptotic sample complexity for adaptively-weighted MPLE . . . 159
4.3.4 Proof of Theorem 4.1.4: Generalization bound for learning the joint distribution . . . 161
4.3.5 Proof of Proposition 4.3.3: Modes of the strongly ferromagnetic Ising model . . . 168
4.3.6 Proof of Proposition 4.1.4: k-Gibbs sampler can reach the mode fast . . . 170
4.3.7 Proof of Proposition 4.1.5 independent parallel sampling stuck in bad samples . . . 171
4.3.8 Proof of Corollary 4.1.1: Separation between N-Gibbs sampler and independent parallel sampling . . . 173
4.3.9 Background and proofs of Proposition 4.1.2 and Proposition 4.1.3: on the expressive power of Transformers for implementing sequence-to-sequence Markov chains in parallel175

4.3.9.1 Technical setup and proofs . . . 175
4.3.9.2 Connection to prior works in GMLM . . . 177
4.3.10 Regularity conditions for asymptotic behavior of M-estimators . . . 179
4.3.11 Convexity of pseudolikelihood for Ising models . . . 180
4.4 Related works . . . 180
4.5 Conclusion . . . 181

5 Conclusion and future directions **182**
5.1 Summary of contributions . . . 182
5.1.1 The mechanics of feature learning in Transformers (Chapter 2) . . . 182
5.1.2 Verifier-guided inference algorithms with backtracking (Chapter 3) . . . 182
5.1.3 Co-designing inference and training algorithms (Chapter 4) . . . 183
5.2 Outlook on future research . . . 183

Chapter 1

Introduction

Language models have achieved remarkable performance across many domains, and are important building blocks of many user-facing applications such as chatbots and coding assistants. However, they still fail at seemingly simple tasks, and moreover, they are brittle in many subtle ways that the research community do not fully understand. In fact, these challenges are not new to language models — for many classic machine learning problems, researchers have developed heuristic algorithms which work well in practice but lack provable guarantee. In these cases, an effective approach to understand when these algorithms work and improve them is to mathematically model the generative process of the data based on realistic assumptions on the key structures. By modeling these structures, researchers can analyze existing heuristic algorithms and design principled innovative algorithmic solutions (Moitra, 2018). Drawing inspirations from these past advancements in the field, I believe an important step towards understanding and improving modern language models is to deepen our understanding on what structural assumptions are reasonable for modeling their training data.

While deep learning theory often assumes that the data is generated by simple generic distributions such as Gaussian or boolean functions, I believe that an important next step for deep learning theory is to focus on ***finer-grained structural assumptions which more closely track real data***. Investigating these more realistic structures enables further progress towards mathematically reasoning about practical deep learning phenomena. For example, consider language structures — modern language models are typically pretrained on massive language data. If we model the pretraining data using generic data distributions such as Gaussian, we can derive generalization bounds on the loss. However, since the data distributional assumptions abstract away language-specific structures, the result cannot answer the more practically-relevant question of what values of the loss suffices for the model to learn certain grammatical rules of English, or to generate text following a coherent topic. Theoretically answering questions of the latter type requires (1) finding a mathematical model of the data distribution which faithfully reflects some aspects of the syntactic or semantic structures underlying real language data, (2) studying how these structures are learned by neural networks during training, and (3) designing inference algorithms for generating new samples consistent with these structures. In this thesis, I present several progresses in my research towards these goals.

How do we model some structures of interest in typical training data?

The massive language model pretraining data can be modeled at different levels of abstractions, leading to various trade-offs. The above paragraph discusses some trade-offs if we use generic distributions such as Gaussian. On the other hand, if we avoid abstracting away any detail by viewing the data "*as is*", i.e. a fixed list of documents, without reasoning about their generative processes, then we can make concrete observations based on a particular dataset, but may miss out on some systematic principles which are transferrable across different datasets. Different from the above perspectives, my research focuses on a middle ground: data distributions which reflect some key properties of language structures (such as syntax and semantics) that are generally present across many language datasets: for example, in (Li & Risteski,

2021) we consider probabilistic context-free-grammars (PCFGs), in (Wen et al., 2023; Botta et al., 2025; Rohatgi et al., 2025) we consider a special case of PCFGs called Dyck grammars (Schützenberger, 1963), and in (Li et al., 2023), we consider topic models (Blei et al., 2003). Based on these data distributions, my research studies how neural networks learn their key structures.

How does the model capture these structures after training?

Neural language models learn many language structures. Yet, these structures were not explicitly enforced by the model as rule-based constraints; instead, we train the model by optimizing its weight parameters to fit training samples, through self-supervised objectives. Thus, the following prerequisite question naturally arises: for a language-structured data distribution mentioned above, *is the model sufficiently expressive to fit a generative process of the distribution*?

My research shows that the answer depends subtly on the design choices about the model architecture. For example, when the data distribution is a PCFG, we prove that a common type of autoregressive models (even with a bounded lookahead window) are insufficiently expressive for accurately parsing PCFGs, and by contrast, we provide explicit constructions based on bidirectional models that can represent the max-likelihood parse of any given PCFG (Li & Risteski, 2021). The key intuition underlying this distinction is that bidirectionality increases the expressivity by allowing the prediction at each position to be based on the full sequence, rather than just a prefix. However, for some harder tasks, even bidirectionality does not guarantee sufficient expressivity: we prove in (Li et al., 2024b) that bidirectional models based on Transformers (Vaswani et al., 2017) do not have sufficient expressivity for representing parallel-decoding Markov chain transitions (which are typical in discrete diffusion language models), and as a result, their mixing times for sampling certain multi-modal distributions are large. [1] By studying the expressivity of common neural architectures for representing solutions that encode certain structures, we identify some theoretical limitations of these architectures, and we hope they will inspire future works which improve model architectures to address these limitations.

In fact, expressivity is not the only factor that determines what structures language models capture after training. *When the model is sufficiently expressive for representing some optima of the loss on certain data distributions, does the training dynamics typically converge to these optima?*

My research shows that for different types of structures in the data distribution, the "degree of freedom" of the training dynamics can be quite different. For some data distributions (such as topic models (Blei et al., 2003)), the training dynamics typically converges to some optima of the loss in a way that the trainable parameters of the model learn to encode the key structure in the data in an intuitive way. To see this, in (Li et al., 2023), we break down the training dynamics into two naturally-separated stages, and characterize the optima in each stage. On the other hand, for some other data distributions (such as the Dyck grammar (Schützenberger, 1963)), the space of optima learned by the model is very rich and does not "uniquely" or "intuitively" encode the structures that define the data distribution. More concretely, in (Wen et al., 2023), we theoretically characterize the sufficient and necessary conditions for model parameters to represent the (exact or approximate) optima of the loss on Dyck grammar distributions, and found that the set of optima encompasses qualitatively diverse solutions. In both cases, experiments validate our theory.

Given the (sometimes large) space of solutions which the model may learn during training, *how do we guide the training dynamics towards learning more generalizable solutions which better capture the key structures in the data?*

Our theory suggests that even though different solutions may simultaneously correspond to the *global optima* of the (in-distribution) training loss, different solutions may differently encode the structure in the data, and generalize differently under natural out-of-distribution scenarios. Correspondingly, we propose theory-inspired regularization techniques to guide the training towards learning features that better align with the groundtruth structures in the data distribution. In the case when the data distribution is a topic model (Blei et al., 2003), our regularization encourages the learned features to preserve symmetries in the topic model. In the case when the data distribution is the Dyck grammar (Schützenberger, 1963),

[1] More details are in Chapter 4.

our regularization encourages the effects of matching brackets to correctly cancel out in the latent space representations, which leads to higher length-generalization accuracy.

Chapter 2 covers more details of my research on the interaction between data and training, focusing on understanding how the model learns to capture the key structures in the data during training.

What inference algorithms best leverage these learned structures?

After language models are trained, assuming they (possibly imperfectly) captured the language structures studied above, *what inference algorithms allow them to generate samples which align well with these learned structures?* My research makes progress towards improving the following two types of inference algorithms:

Incorporating a verifier to assist the language model generator Even for a language model which approximately captured certain language structures, when it generates new sequences, it may still make mistakes which violate these structural constraints. Researchers have developed inference-time algorithms by incorporating a *verifier* which can score (complete or partial) generations of the language model (Cobbe et al., 2021; Nakano et al., 2022). Though a flurry of recent papers consider "scaling laws" of natural verifier-aided inference-time algorithms, the *value of a verifier*—and the *relationship it needs to have to the generator* is still not well understood. My research (Botta et al., 2025) proves that the incorporation of process verifiers is necessary for certain constrained generation tasks (both information-theoretically and computationally), and shows empirically (on Dyck grammar and Python code generation tasks) that backtracking is a surprisingly effective rejection sampling strategy when a process verifier is available. Moreover, in (Rohatgi et al., 2025) we generalize a classic algorithm in the approximate counting and sampling community of theoretical computer science, namely the *Sinclair-Jerrum random walk* (Sinclair & Jerrum, 1989), and apply it to process verifier-guided generation. The inference algorithm we propose is theoretically principled: when the verifier is imperfect (which is true in most practical cases), our algorithm provably mitigates error amplification as the sequence length grows. Empirical observations based on synthetic data and real language tasks verify our theory. Chapter 3 covers more details of my research on the interaction between data and inference, focusing on establishing a theoretical framework for reasoning about verifier-assisted language generation, and identifying *backtracking* to be a key component in verifier-assisted language generation algorithms.

Non-autoregressive parallel generation Besides autoregressive language models which fit the next-token probabilities, a promising line of works (Ghazvininejad et al., 2019; Gu & Kong, 2021; Savinov et al., 2022; Lou et al., 2024; Sahoo et al., 2024; Kim et al., 2025) develop non-autoregressive language models which are trained to fit conditional probabilities for parts of the sequence (by applying a mask), conditioned on the rest. These conditionals will be used as oracles for running a Markov Chain to generate samples. Note that at inference time, with each forward pass through the decoder layers, these models predict not just one next token, but multiple tokens in parallel. Such parallelism enabled them to generate texts much faster than autoregressive models. However, note that the conditionals learned by these models do not necessarily form a consistent joint distribution. Thus, it is not well-understood what structures in the data can be efficiently learned and accurately sampled by these models. Towards elucidating these questions, my research (Li et al., 2024b) proves that the sample-efficiency of *training* masked language models is closely connected to the computational-efficiency of *inference*, and in particular, strong cross-position dependency in the data distribution is a challenge for both training and inference. Moreover, we prove that masking more tokens when training these conditionals is more sample-efficient. Finally, we theoretically identify a limitation of Transformers for parallel decoding: they enforce conditional product distributions, which can prevent fast mixing at inference time. Chapter 4 covers more details of my research on the interaction between training and inference, focusing on revealing a deep connection between training efficiency and inference efficiency. Moreover, the same quantity which governs both *training* and *inference* efficiency bounds is in fact a property of the *data* distribution. These findings close the loop: we establish a theoretical framework for mathematically reasoning about the data-training-inference interaction for this type of parallel-efficient language modeling approaches.

Chapter 2

How Transformers learn simple linguistic structures

During training, modern language models parameterized by deep neural networks learn a large set of *features*, i.e. real-valued vector representations of important signals in the input (LeCun et al., 2015; Goodfellow et al., 2016). These features, automatically learned by optimizing the empirical risk minimization (ERM) objective using gradient descent-based optimization algorithms on the neural network parameters, have been shown to outperform handcrafted features in many downstream applications (Krizhevsky et al., 2012; Devlin et al., 2019). However, some of these automatically learned features are *spurious features* harmful for the downstream task performance (Tu et al., 2020). Can we understand what features are learned, and guide the training process towards learning more robust features?

Existing theory on the training process of neural networks are mostly based on the neural tangent kernel (NTK) (Jacot et al., 2018), but the NTK theory cannot explain the feature learning process (Li et al., 2020). Recent theories have begun developing understanding of the training process that incorporates feature learning (Allen-Zhu & Li, 2020), including our own work on understanding some common self-supervised learning objectives (Pokle et al., 2022), but they consider very simple neural network model architectures or data distributions. Since different model architectures possess different inductive biases which benefit different data distributions (for example, Transformers robustly outperform prior architectures on natural language tasks (Vaswani et al., 2017), whereas convolutional neural networks are still leading many image benchmarks (Liu et al., 2022b)), we think it is important to develop the theoretical toolbox for understanding the feature learning process while incorporating the interplay of *model architectures* and *data distributions*. The key question we ask is: what properties of the data distribution are captured by which part of the neural network model architecture?

In Section 2.1 (based on Li et al. (2023)), we develop mechanistic understanding of how a simple, 1-layer Transformer learns topic structure. In Section 2.2 (based on Wen et al. (2023)), we prove that even on simple (context-free) grammars, even for small (2-layer) Transformers, the solution space is very rich and does not “uniquely and interpretably” encode grammatical structure. In both cases, our research covers theory and experiments on the types of features that transformer-based language models learn through standard pretraining, and propose theory-inspired regularization techniques to guide the Transformer towards learning features that better align with the groundtruth structures in the data distribution.

2.1 How Transformers learn topic structure: towards a mechanistic understanding

While the successes of transformers across many domains are indisputable, accurate understanding of the learning mechanics is still largely lacking. Their capabilities have been probed on benchmarks which include a variety of structured and reasoning tasks—but mathematical understanding is lagging substantially behind. Recent lines of work have begun studying representational aspects of this question: that is, the size/depth/complexity of attention-based networks to perform certain tasks. However, there is no guarantee the learning dynamics will converge to the constructions proposed. In our paper, we provide fine-grained mechanistic understanding of how transformers learn "semantic structure", understood as capturing co-occurrence structure of words. Precisely, we show, through a combination of mathematical analysis and experiments on Wikipedia data and synthetic data modeled by Latent Dirichlet Allocation (LDA), that the embedding layer and the self-attention layer encode the topical structure. In the former case, this manifests as higher average inner product of embeddings between same-topic words. In the latter, it manifests as higher average pairwise attention between same-topic words. The mathematical results involve several assumptions to make the analysis tractable, which we verify on data, and might be of independent interest as well.

The transformer architecture (Vaswani et al., 2017) is a critical building block of many leading approaches to natural language processing (Devlin et al., 2019; Brown et al., 2020), and other domains such as vision (Dosovitskiy et al., 2021) and protein structure prediction (Jumper et al., 2021). While the NLP community has produced a large body of work on probing and visualizing trained networks (Hewitt & Manning, 2019; Clark et al., 2019; Tenney et al., 2019; Kovaleva et al., 2019), we still have little formal understanding of the mechanisms by which transformers, trained with simple gradient-descent based algorithms, learn from their training data. The challenge is that the training dynamics are non-trivial, even for relatively simple structured data distributions, and even for simple (e.g. 1-layer) transformers.

In particular, we study *semantic structure*, as understood through the lens of *co-occurrences* of words, and their topical structure. Precisely, if we fit topics to a real-life corpus like Wikipedia using a *Latent Dirichlet Allocation* (LDA, Blei et al., 2003) model, we find a pretrained BERT model produces token embeddings that are more similar (in terms of inner product or cosine similarity) if they belong to the same topic, and more different if they belong to different topics (see e.g. Figure 2.3).

Inspired by these observations, we study LDA-generated data as a sandbox to understand—both through experiments on such synthetic data, and theoretical results—the process by which the embeddings and attention learn the topical structure. We find that the above observations from Wikipedia data are even more pronounced on synthetic LDA data. Moreover, we mathematically prove why such structure arises by analyzing a simplified *two-stage training dynamics* for a single-layer transformer trained under the masked language modeling objective. We also verify the two-stage nature of training dynamics obtains for a wide variety of optimizers and hyperparameter settings. [1]

2.1.1 Technical setup

Topic models For our theoretical analysis, in order to have a well-defined notion of a "ground truth", we will consider data distribution generated by a topic model consisting of T topics $\{1, \cdots, T\}$ and Tv words $\{1, \cdots, Tv\}$. We will in fact, consider a special case of an LDA (Latent Dirichlet Allocation) model (Blei et al., 2003). Precisely, each document $\boldsymbol{w}$ is a sequence of words $w_1, \cdots, w_N$, and is generated by: [2]

1. Randomly choose τ distinct topics $t_1, \cdots, t_\tau$ from $[T]$.
2. For $n \in [N]$:

(a) Randomly choose a topic t from $\{t_1, \cdots, t_\tau\}$.

[1] Code is released at https://github.com/YuchenLi01/transformer_topic_model_LDA

[2] Our theoretical results crucially depend on all topics being disjoint, i.e. they do not share common words. It is not crucial that the words in the same topic all have the same probabilities. Allowing these probabilities to be different would lead to results of similar flavor, but complicates the notation.

(b) Randomly choose w_n from $\{(t-1)v+1, \cdots, tv\}$.

Note, under this data distribution, each word belongs to exactly one topic, and different topics do not share common words.

Definition 2.1.1 (Topic-word indicator)**.** *A word i belongs to topic t (denoted as $i \in t$) if $i \in \{(t-1)v+1, \cdots, tv\}$. Correspondingly,* $\texttt{topic}(i) \coloneqq \lceil \frac{i}{v} \rceil$

Let $\mathcal{D}_{\boldsymbol{w}}$ denote the distribution of documents following the above generative process. Furthermore, for each document $\boldsymbol{w}$, let $\boldsymbol{X} \in \{0,1\}^{(Tv+1)\times N}$ denote its *one-hot* encoding, in which $X_{ij} = 1$ if $w_j = i$, and 0 otherwise. Analogous to $\mathcal{D}_{\boldsymbol{w}}$, let $\mathcal{D}_{\boldsymbol{X}}$ denote the distribution of document one-hot encodings.

To simplify our theoretical analysis, we consider the *infinitely-long-document* setting, such that within each document, the empirical token distribution is equal to the groundtruth token distribution:

Assumption 2.1.1 (Infinitely-long documents)**.** *Each document $\boldsymbol{w}$ consists of exactly τ topics $\{t_1, \cdots, t_\tau\}$. Moreover, for each word $i \in \{1, \cdots, Tv\}$ in the vocabulary, its empirical probability in the document*

$$\boldsymbol{p}_{\boldsymbol{w}}(i) = \frac{\sum_{n=1}^{N} \mathbb{1}_{w_n = i}}{N} = \begin{cases} \frac{1}{\tau v}, & \textit{if } i \in \cup_{j=1}^{\tau} t_j \\ 0, & \textit{otherwise} \end{cases}$$

In our synthetic data experiments, we use a finite N and generate data using an LDA model (Blei et al., 2003) which allows for slightly more variability—and demonstrates that our results are robust to changes in the setting. Detailed experimental setup is described in Section 2.1.5.

Training objective Given data following the distribution defined in Section 2.1.1, we train a transformer network using the masked language modeling objective (Devlin et al., 2019). We first define the token `[MASK]` $= 0$ in addition to the words $\{1, \cdots, Tv\}$ of the topic model. Three constant probabilities $p_m, p_c, p_r \in (0,1)$ specify the masking scheme:

1. For the original document $\boldsymbol{w} = w_1 \cdots w_N$, first randomly choose a set of masked indices $M(\boldsymbol{w}) \subset [N]$ such that $\forall i \in [N]$, with probability p_m, $i \in M(\boldsymbol{w})$.
2. Define the masked document $\tilde{\boldsymbol{w}} = \tilde{w}_1 \cdots \tilde{w}_N$ such that for each $i \in [N]$,
   (a) If $i \notin M(\boldsymbol{w})$, then $\tilde{w}_i = w_i$.
   (b) If $i \in M(\boldsymbol{w})$, then $\tilde{w}_i = \begin{cases} w_i, \text{with probability } p_c \\ \text{random word in } [Tv], \text{with probability } p_r \\ \texttt{[MASK] = 0}, \text{with probability } 1 - p_c - p_r \end{cases}$

Given a document $\boldsymbol{w}$ and its masked version $\tilde{\boldsymbol{w}}$, the model f_θ (parameterized by θ) observes $\tilde{\boldsymbol{w}}$ and is trained to predict the original words at the masked positions M. More formally, given the one-hot encoding of the masked document $\tilde{\boldsymbol{X}}$, and the model prediction $\hat{\boldsymbol{X}} = f_\theta(\tilde{\boldsymbol{X}}) \in \mathbb{R}^{(Tv+1)\times N}$, letting $\boldsymbol{X}_{:j}$ denote the j-th column of matrix $\boldsymbol{X}$, for some loss function $l(\cdot,\cdot) \to \mathbb{R}$, the training objective is $\min_\theta L(\theta)$ for

$$L(\theta) = \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}} \mathbb{E}_M \frac{1}{|M|} \sum_{j\in M} l(f_\theta(\tilde{\boldsymbol{X}})_{:j}, \boldsymbol{X}_{:j}) \tag{2.1}$$

Motivated by the empirical success of applying weight decay to training transformers, we also consider a regularized version of the above masked language modeling objective. For L_2-regularization [3] with parameter $\lambda > 0$:

$$L_{\text{l2reg}}(\theta) = L(\theta) + \lambda \|\theta\|_2^2 \tag{2.2}$$

[3] When θ is a vector, L_2-regularization penalizes $\|\theta\|_2$. When θ is a matrix, the correct norm to regularize is $\|\theta\|_F$.

Our theoretical analysis uses the squared loss: given a prediction vector $\boldsymbol{x} \in \mathbb{R}^d$ and an one-hot label vector $\boldsymbol{y} \in \{0,1\}^d$ in which $y_i = 1$ and $\forall j \neq i, y_j = 0$

$$l(\boldsymbol{x}, \boldsymbol{y}) \coloneqq l_{\text{sq}}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{2.3}$$

Our experiments additionally study the cross entropy loss:

$$l(\boldsymbol{x}, \boldsymbol{y}) \coloneqq l_{\text{ce}}(\boldsymbol{x}, \boldsymbol{y}) = -\log \frac{\exp{(\boldsymbol{x}_i)}}{\sum_{j=1}^{d} \exp{(\boldsymbol{x}_j)}} \tag{2.4}$$

Remark 2.1.1. *We give results for both types of loss functions because the cross-entropy loss, albeit practically more commonly used, is theoretically less convenient. Concretely, it involves the softmax operation which is invariant under addition by the same constant in each dimension (implying that the optimal logits are not necessarily unique); moreover, the optimal logits are often at infinity. By contrast, with squared loss, the set of optima is more easily characterized using some finite-valued closed form expressions.*

Empirically, we will show (in Section 2.1.5) that the conclusions in our theoretical analyses hold for both the cross-entropy loss and the squared loss, as well as with variants of the training algorithm like SGD and Adam.

Transformer network architecture To theoretically reason about the role played by the embedding layer and the self-attention layer, we consider a one-layer, single-head transformer model (Vaswani et al., 2017) with the simplification that the residual connection and normalization layers are removed. Precisely:

$$f(\boldsymbol{Z}) = \boldsymbol{W}^{\text{pred}}(\boldsymbol{W}^V \boldsymbol{Z})\sigma(\frac{(\boldsymbol{W}^K \boldsymbol{Z})^\top(\boldsymbol{W}^Q \boldsymbol{Z})}{\sqrt{d_a}}) + \boldsymbol{b}^{\text{pred}}$$

$\boldsymbol{Z} \in \mathbb{R}^{d \times N}$ is the input representation. d is the embedding dimension. $\boldsymbol{W}^{\text{pred}} \in \mathbb{R}^{V \times d}$ and $\boldsymbol{b}^{\text{pred}} \in \mathbb{R}^V$ are the prediction head weights and biases. V is the vocabulary size. In our masked language modeling setting (Section 2.1.1), $V = Tv + 1$. $\boldsymbol{W}^V \in \mathbb{R}^{d \times d}$ is the value matrix weight. $\sigma : \mathbb{R}^{N \times N} \mapsto (0,1)^{N \times N}$ is the column-wise softmax operation, such that $\sigma(A)_{ij} = \frac{\exp{(A_{ij})}}{\sum_{l=1}^{N} \exp{(A_{lj})}}$. d_a is the attention head size. $\boldsymbol{W}^K \in \mathbb{R}^{d_a \times d}$ is the key matrix. $\boldsymbol{W}^Q \in \mathbb{R}^{d_a \times d}$ is the query matrix. Let $A(\boldsymbol{Z})$ denote the attention weights:

$$A(\boldsymbol{Z}) \coloneqq \sigma\left(\frac{(\boldsymbol{W}^K \boldsymbol{Z})^\top(\boldsymbol{W}^Q \boldsymbol{Z})}{\sqrt{d_a}}\right) \in (0,1)^{N \times N} \tag{2.5}$$

In our setting, the input $\boldsymbol{Z}$ is the embedding of the masked document, i.e. $\boldsymbol{Z} = \boldsymbol{W}^E \tilde{\boldsymbol{X}}$ for some embedding weights $\boldsymbol{W}^E \in \mathbb{R}^{d \times (Tv+1)}$. Moreover, following empirical best practice (Press & Wolf, 2017) and standard implementation in (Wolf et al., 2020), we weight-tie the prediction head weight $\boldsymbol{W}^{\text{pred}}$ and the embedding weight $\boldsymbol{W}^E$:

$$f(\tilde{\boldsymbol{X}}) = {\boldsymbol{W}^E}^\top \boldsymbol{W}^V \boldsymbol{W}^E \tilde{\boldsymbol{X}} A(\boldsymbol{W}^E \tilde{\boldsymbol{X}}) + \boldsymbol{b}^{\text{pred}} \tag{2.6}$$

In part of our theoretical analysis (in Section 2.1.4) and experiments (in Section 2.1.5), we freeze *one-hot* word embeddings, to study the mechanism that self-attention represents the topic structures without the aid of trained token embeddings. That is, set $d = Tv + 1$ and $\boldsymbol{W}^E = I$:

$$f(\tilde{\boldsymbol{X}}) = \boldsymbol{W}^V \tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}}) + \boldsymbol{b}^{\text{pred}} \tag{2.7}$$

The positional encoding at the input is removed because the position information of a word in a document is irrelevant to the topic model defined in Section 2.1.1.

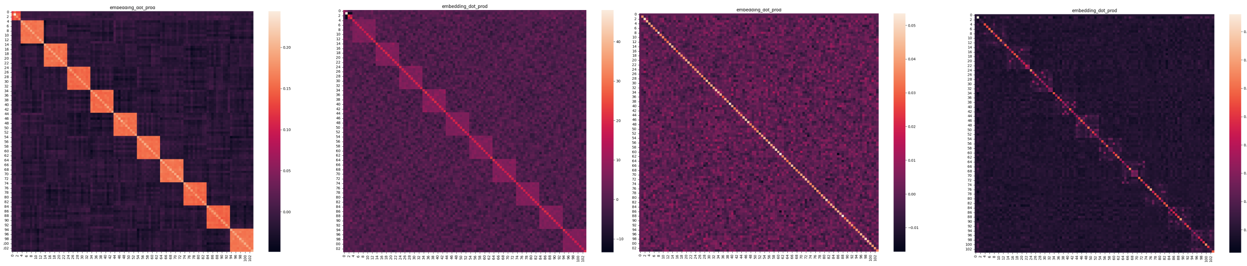


Figure 2.1: Embedding weight dot product of models trained on synthetic topic modeling data (Section 2.1.5.1). The four plots correspond to different combinations of loss function and optimizer: (left to right) cross-entropy with SGD, cross-entropy with Adam, squared loss with SGD, squared loss with Adam, all using learning rate 0.01. The block-wise pattern verifies our theory in Section 2.1.3. The 10 blocks correspond to the 10 topics in the data distribution in Section 2.1.1. In particular, a diagonal pattern is a special case of the block-wise optima that we prove (see Theorem 2.1.1).

2.1.2 Overview of results

We focus on understanding the optimization dynamics of transformers in a simple sandbox: a single-layer transformer trained on (synthetic) data following a topic model distribution—and validate that our results robustly transfer to real data (Wikipedia WikimediaFoundation, 2023). We show that topic structure can be encoded both in the embedding layer, and in the attention mechanism of the network. Moreover, even if one of these components is not trained (i.e. handicapped), the other can "compensate" for it.

Theoretically, we characterize precisely how the topic structure is learned in the two extremal cases: when the attention mechanism is frozen to be uniform, and the only model parameters that are trained are the token embeddings; and when the token embeddings are frozen to be one-hot vectors, and the attention parameters (the key, query, and value matrices) are trained. We empirically verify our characterization on synthetic LDA-generated data, and also show that on real Wikipedia data, topic structure is learned both in the embeddings, and the attention mechanism.

2.1.2.1 Topic structure is encoded in token embeddings

In the first extremal case, we analyze the optima when we solely train the embedding layer. Precisely, we show that even when we freeze the attention scores to be uniform and all other elements of the transformer are set to identity, the model can still achieve near optimal loss by "encoding" the topic structure in the embedding weights:

Theorem (Optimal word embedding, informal)**.** *Suppose the training data follows a topic model data distribution, and the transformer has trainable embedding layer, frozen (uniform) attention scores, and all other components set to identity. Then, the optimal embedding layer of a single layer transformer is such that the inner product of the embeddings of a pair of words is larger when the words belong to the same topic, and smaller when they belong to different topics.*

Intuitively, this result states that words of the same topic, after training, have more similar embeddings than words of different topics. In this sense, the embedding layer captures the topic structure. We also empirically show (Section 2.1.5 and Figure 2.1) that this phenomenon is robust to differences in loss function and optimization method. See Section 2.1.3 for the formal theorem and Section 2.1.7.2 for the proof.

2.1.2.2 Topic structure is encoded in self-attention

In the second extreme, we study the behavior of the self-attention in a transformer trained on a topic modeling distribution, without the aid of trained token embeddings — i.e. when we use hard-coded, *one-hot* embeddings. The attention weight matrices $\boldsymbol{W}^K$, $\boldsymbol{W}^Q$, and $\boldsymbol{W}^V$ are initialized to near-zero matrices. To

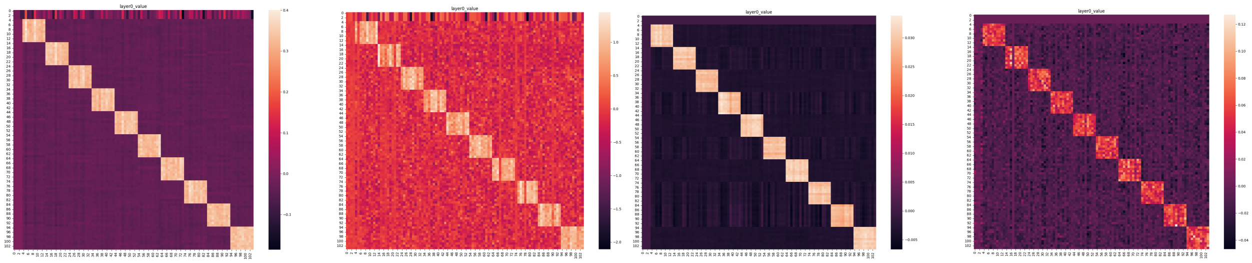


Figure 2.2: Convergence point of trained $\boldsymbol{W}^V$ (with L_2-regularization) when freezing uniform attention weights and one-hot word embedding. The four plots correspond to different combinations of loss function and optimizer. (Left to right) cross-entropy with SGD, cross-entropy with Adam, squared loss with SGD, squared loss with Adam, all using learning rate 0.01. The block-wise pattern verifies our theory in Section 2.1.4.2. The 10 blocks correspond to the 10 topics in the data distribution. Results are qualitatively similar without L_2-regularization, or if we train $\boldsymbol{W}^K$ and $\boldsymbol{W}^Q$ instead of freezing them (see Section 2.1.5.2).

make the analysis feasible, we break down the training process into two separate stages, and characterize the optima in each stage. In the *first stage*, the attention is frozen to be uniform, and the matrix $\boldsymbol{W}^V$ is trained. In the *second stage*, the matrix $\boldsymbol{W}^V$ is frozen to the optimal value from the first stage, and the optimal attention weights is analyzed. Intuitively, such a two-stage approximation is reasonable, because in the initial stages of training, the gradients for the value matrix are much larger than those for the key and query matrices (see Section 2.1.6). While this is an approximation, this two-stage phenomenon can be observed empirically for a variety of hyperparameter settings (see Section 2.1.4.1 and in particular Figure 2.4). We also provide empirical evidence that the optima characterized in our analysis closely track the actual convergence points of models.

In brief, the self-attention function is $\text{Attn}(Z) \coloneqq \boldsymbol{W}^V \boldsymbol{Z} A(\boldsymbol{Z})$ in which $A(\boldsymbol{Z})$ denotes the attention weights, and $\boldsymbol{W}^V$ is the value matrix weight. Intuitively, $A(Z)_{ij}$ is the importance of the i-th word for predicting the j-th word, and $\boldsymbol{W}^V$ is aggregates the word embeddings in a sentence, weighted by the attention weights $A(\boldsymbol{Z})$. The formal definition of the model architecture is in Section 2.1.1.

Optimal W^V in Stage 1 We characterize the optimal $\boldsymbol{W}^V$ in the initial stage of training: $\boldsymbol{W}^V$ will learn a block-wise structure (see Figure 2.2), in which each block corresponds to a topic:

Theorem (Optimal $\boldsymbol{W}^V$, informal)**.** *Suppose the training data follows a topic model data distribution, the token embeddings are frozen to be one-hot vectors, and attention scores are frozen to be uniform. Then, under mild L_2 regularization, the optimal $\boldsymbol{W}^V$ for the masked language modeling objective has block-wise structure, namely the (i, j)-th entry of $\boldsymbol{W}^V$ is on average larger when the tokens i and j belong to the same topic, and on average smaller when the tokens i and j belong to different topics.*

For the formal theorem statement, see Section 2.1.4. The proof is deferred to Section 2.1.7.3. We also empirically show (Section 2.1.5 and Figure 2.2) that this phenomenon is robust to differences in training loss and optimization method.

Optimal attention weights in Stage 2 For the second stage of the training dynamics, we assume $\boldsymbol{W}^V$ is frozen to the optimal value in the first stage, and train the attention weights.

Theorem (Optimal attention weights, informal)**.** *Suppose a single layer transformer is trained on a topic model data distribution, and $\boldsymbol{W}^V$ is frozen to the block-wise first-stage optima. Then, the optimal attention weight for the masked language modeling objective is such that on average: a convex combination of* same-word *attention and* same-topic-different-words *attention should be relatively large, compared to* different-topic *attention.*

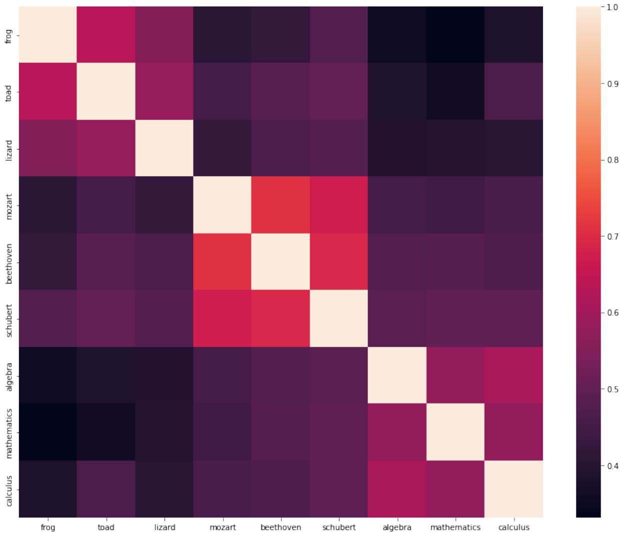


Figure 2.3: For a BERT model pre-trained on Wikipedia corpus, the cosine similarity of the word embeddings encodes topical structures, i.e. it is larger if the two words belong to the same topic, and smaller if they belong to different topics. This phenomenon is more pronounced for words that are very likely only under a few topics. In this figure, the nine words fall into three topics: {frog, toad, lizard} are animals, {mozart, beethoven, schubert} are musicians, and {algebra, arithmetic, calculus} are mathematical concepts.

For the formal assumption and theorem statements, see Section 2.1.4. The proof is deferred to Section 2.1.7.7.

We empirically show (in Section 2.1.5) that even when the all the self-attention weight matrices are *jointly* trained (instead of trained with the two-stage process described), the behavior of attention weights still follows the relations that the above theorem describes.

2.1.2.3 Empirical results

We provide empirical evidence that the main conclusions in our theoretical findings remain robust even under settings that are more complex and realistic than our theoretical setup, and under variations of the training algorithm and loss. For example, we also test on synthetic data using a Latent Dirichlet Allocation (LDA) topic model (Blei et al., 2003) instead of our simplified topic modeling distribution; finally, we report results for a model pre-trained on the Wikipedia textual corpus, and discuss the connections with our conclusions derived in the synthetic setting. We describe detailed experimental setup and results in Section 2.1.5.

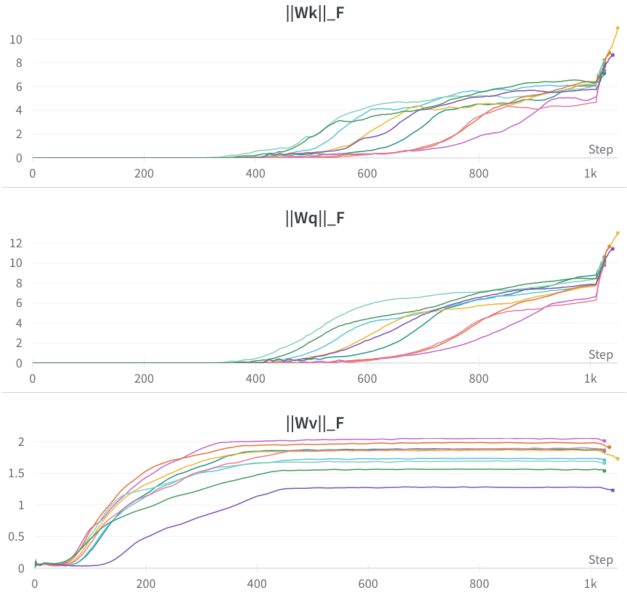


Figure 2.4: Two-stage learning dynamics of a single-layer transformer trained on LDA data distribution. All weight matrices are initialized to random matrices near zero, and *simultaneously trained*. The learning dynamics naturally exhibits a *two-stage* phenomenon: in **Stage 1** (steps 0-400), the norms of the key matrix (W^K, top) and the query matrix (W^Q, middle) stay close to 0, while the norm of the value matrix (W^V, bottom) increases significantly. In **Stage 2** (steps 400-1000), the norms of W^K and W^Q start increasing significantly, while the norm of W^V stays relatively flat. Different curves in the figure correspond to different settings of the hyperparameters as well as different runs in each setting. (See Section 2.1.6 for more details.)

2.1.3 Topic Structure Can Be Encoded in Token Embeddings

The first result shows that, under the topic model data distribution, even if we freeze the self-attention to be uniform, the embedding layer can encode the topic structure. Precisely:

Theorem 2.1.1 (Optimal token embedding)**.** *Suppose the data distribution follows the topic modeling assumption in Section 2.1.1 and Assumption 2.1.1. Suppose we train a single layer transformer given by* (2.6) *with* $\boldsymbol{W}^K = 0, \boldsymbol{W}^Q = 0, \boldsymbol{W}^V = I$ *and* $\forall i, \boldsymbol{b}_i^{pred} = -\frac{p_m p_r}{(1-(1-p_c)p_m)Tv}$*, under the masked language modeling objective* ((2.1)) *with the squared loss* ((2.3))*. Then, there exist constants* $u_0, \cdots, u_{Tv} \in \mathbb{R}$ *such that the optimal word embedding weight* $\boldsymbol{W}^E$ *and* $\boldsymbol{E} \coloneqq {\boldsymbol{W}^E}^\top \boldsymbol{W}^E$ *satisfy:*

1. *The 0-th row of* $\boldsymbol{E}$ *satisfies:*

   (a) $\boldsymbol{E}_{00} = -\left(\frac{1}{p_m(1-p_c-p_r)} - 1\right) \cdot u_0$

   (b) $\forall t \in [T], \sum_{l \in t} \boldsymbol{E}_{0l} = u_0 v$

2. *The 0-th column of* $\boldsymbol{E}$ *satisfies* $\forall i \in \{1, \cdots, Tv\}$*:*

   (a) $\boldsymbol{E}_{i0} = -\left(\frac{1}{(1-p_c-p_r)p_m} - 1\right) u_i$

3. $\boldsymbol{E}_{ij}$ ($\forall i, j \in \{1, \cdots, Tv\}$) *satisfy:*

   (a) $\sum_{l \in \texttt{topic}(i)} \boldsymbol{E}_{il} = u_i v + \frac{1}{1-(1-p_c)p_m}$

   (b) $\forall t \in [T]$ *such that* $\texttt{topic}(i) \neq t$, $\sum_{l \in t} \boldsymbol{E}_{il} = u_i v$

Remark 2.1.2. *Point 3 is the important one among the list of conclusions. The way to read the theorem is that, among the entries of an optimal* $\boldsymbol{E}$*: for i and j corresponding to the indices of tokens of the* ***same topic****,* $\boldsymbol{E}_{ij}$ *is (on average) larger, meaning that the embeddings of same-topic tokens are more similar; for i and j corresponding to* ***different topics****,* $\boldsymbol{E}_{ij}$ *is (on average) smaller, meaning that the embeddings of different-topic tokens are less similar. In particular, when the constants* $u_0, \cdots, u_{Tv}$ *are all zero, then the above larger-vs-smaller difference becomes a positive-vs-zero difference, which we roughly observe in practice.*

Remark 2.1.3. *Intuitively, the setting of the bias* $\boldsymbol{b}^{pred}$ *is used to "denoise" the masked sequence, i.e. to subtract the probability caused by filling in random words in the masking process (described in Section 2.1.1).*

The proof of this theorem is deferred to Section 2.1.7.2.

Proving comparable results under cross-entropy loss ((2.4)) is more challenging considering Remark 2.1.1. However, we empirically show that, such blockwise pattern in $\boldsymbol{E} \coloneqq {\boldsymbol{W}^E}^\top \boldsymbol{W}^E$ tends to exist in a trained model under both the squared loss and the cross-entropy loss, and regardless of whether we (i) train all layers or (ii) only train the embedding layer while freezing all other layers. Moreover, the loss achieved in case (ii) is only slightly worse than in case (i). Finally, we also show (Figure 2.3) that on real data, words that are unambiguous (e.g. "calculus", "Mozart") exhibit a similar pattern as Theorem 2.1.1 states: same-topic words have more similar embeddings, and therefore larger embedding dot products, than different-topic words. Quantitatively, if we only restrict ourselves to words that are unambigious (i.e. likely to be emitted only under few topics), a similar phenomenon can be observed (see Table 2.4).

2.1.4 Topic Structure Can Be Encoded in Self-Attention

Whereas the previous section showed that the token embedding layer can in principle perform the heavy-lifting in learning the topic-modeling distribution, we further show that self-attention *also* can encode the topic structures, when we disallow training the embedding layer. That is, we freeze the token embeddings to be one-hot.

2.1.4.1 The two-stage optimization process of self-attention

While inspecting the training dynamics of this one-layer transformer on the topic modeling data distribution, we observed a roughly *two-stage* process (illustrated by Figure 2.4): with certain initialization and learning rate settings, in **Stage 1**, the key matrix (W^K) and the query matrix (W^Q) stay close to 0, i.e. each position pays a near-uniform attention to all positions in the document, while the norm of the value matrix (W^V) increases significantly. In **Stage 2**, the norm of the the value matrix (W^V) already plateaus, and only after that, do the key and query matrices (W^K and W^Q) start to move.

Thus, while reasoning about the training process of transformers in our data distribution, we take motivation from the above empirical observation of such two-stage process, and consider a corresponding simplification: in Stage 1, the attention is frozen to be uniform, and only $\boldsymbol{W}^V$ is trained; in Stage 2, $\boldsymbol{W}^V$ is frozen, while $\boldsymbol{W}^K$ and $\boldsymbol{W}^Q$ are trained. This simplification is a reasonable proxy for standard training, and we furthermore validate that our theoretical characterizations are robust to standard training, both using SGD and Adam. We provide more discussion on the two-stage optimization process in Section 2.1.6.

2.1.4.2 Optimal W^V given uniform attention

The Stage 1 of optimization process is convex (but not strongly convex) in $\boldsymbol{W}^V$, and we show that the set of minima consist of exactly the set of $\boldsymbol{W}^V$ that exhibits a *block-wise* pattern:

Theorem 2.1.2 (Optimal $\boldsymbol{W}^V$ with mild L_2-regularization when freezing uniform attention)**.** *Suppose the data distribution follows the topic modeling assumption in Section 2.1.1 and Assumption 2.1.1. Suppose we train a single layer transformer given by* (2.7) *with* $\boldsymbol{W}^K = 0, \boldsymbol{W}^Q = 0, \boldsymbol{b}^{pred} = 0$*, under the* L_2*-regularized masked language modeling objective (*(2.2)*) with the squared loss (*(2.3)*). Then,* $\lim_{\lambda \to 0} \arg\min L_{l2reg}(\boldsymbol{W}^V) = \{\boldsymbol{W}^{V*}\}$ *in which* $\boldsymbol{W}^{V*} \in \mathbb{R}^{(Tv+1)\times(Tv+1)}$ *satisfies:*

1. *The 0-th row of* $\boldsymbol{W}^{V*}$*:*

   (a) $\forall j \in \{0, \cdots, Tv\}, \boldsymbol{W}^{V*}_{0j} = 0$

2. *The 0-th column of* $\boldsymbol{W}^{V*}$*:*

   (a) $\forall i \in \{1, \cdots, Tv\}, \boldsymbol{W}^{V*}_{i0} = \frac{c_2 c_3 - c_1 Tv}{c_2^2 + Tv}$

3. $\boldsymbol{W}^{V*}_{ij}$ *(*$\forall i, j \in \{1, \cdots, Tv\}$*):*

(a) $\forall l \notin \texttt{topic}(i),\ \boldsymbol{W}^{V*}_{il} = \boldsymbol{W}^{V*}_{\textit{diff-topic}} := -\frac{c_1 c_2 + c_3}{c_2^2 + Tv}$

(b) $\forall l \in \texttt{topic}(i), \boldsymbol{W}^{V*}_{il} = \boldsymbol{W}^{V*}_{\textit{same-topic}} := \boldsymbol{W}^{V*}_{\textit{diff-topic}} + \frac{c_3}{v}$

in which the constants are:

- $c_1 = \frac{p_r}{(1-p_c-p_r)(1-(1-p_c)p_m)Tv} \in (0, 1)$
- $c_2 = \frac{1}{(1-p_c-p_r)p_m} - 1 \in (0, +\infty)$
- $c_3 = \frac{1}{1-(1-p_c)p_m} \in (1, +\infty)$

Empirically, on our topic model data distribution, the loss achieved by freezing $\boldsymbol{W}^K = \boldsymbol{W}^Q = 0$ and only training $\boldsymbol{W}^V$ is only slightly greater than the loss achieved by training all of them jointly.

Intuitively, this block-wise $\boldsymbol{W}^V$ shows that, while inferring about the words at the masked positions: the model looks at unmasked positions in the document, each unmasked word only contributes to predicting words of the *same topic*, each unmasked word does not contribute to predicting words of *different topics*, and the model implicitly aggregates the topic distribution among the unmasked words, to infer the token distribution in the original document prior to masking.

The proof of this Theorem 2.1.2 is deferred to Section 2.1.7.3. Proving a comparable result under the cross-entropy loss (2.4) is more challenging due to the same reasons outlined in Remark 2.1.1. However, empirically such block-wise $\boldsymbol{W}^V$ shows up for both the cross-entropy loss and the squared loss, as we show in Section 2.1.5.

2.1.4.3 Optimal attention weights

In our analysis on the stage 2 optimization process, we freeze the $\boldsymbol{W}^V$ to be some representative optima from stage 1 (Theorem 2.1.2), and characterize the optimal attention weights by comparing the following three types of attention weights: among the *same words* at different positions, among different words of the *same topic*, and among words of *different topics*.

We mainly consider the type of optimal $\boldsymbol{W}^V$ characterized in Theorem 2.1.2: $\boldsymbol{W}^V$ with uniform blocks (see Figure 2.2). Empirically, the model often approximately converges to these type of pattern (Section 2.1.5).

To formally reason about the behavior of average attention weights, we consider a simplified setting:

Assumption 2.1.2 (Attention pattern)**.** *Following the notation in* (2.5)*, assume that for any masked document* $\tilde{\boldsymbol{w}}$ *with embedding* $\tilde{\boldsymbol{X}}$*,*

$$A(\tilde{\boldsymbol{X}})_{ij} = \begin{cases} c_1, \textit{if } \tilde{w}_i = \tilde{w}_j \\ c_2, \textit{if } \tilde{w}_i \neq \tilde{w}_j \textit{ but } \texttt{topic}(\tilde{w}_i) = \texttt{topic}(\tilde{w}_j) \\ c_3, \textit{if } \texttt{topic}(\tilde{w}_i) \neq \texttt{topic}(\tilde{w}_j) \end{cases}$$

in which $c_2 = \alpha c_3$ *and* $c_1 = \beta c_3$*.*

We note that this family of attention weights is *realizable*, and by symmetricity (among different topics and among the words in the same topic) and convexity (in $A(\tilde{\boldsymbol{X}})$), it is simple to prove that the attention pattern outlined in Assumption 2.1.2 is among the optimal attention patterns.

We will characterize the setting of α and β that minimizes the loss, under the following assumptions:

Assumption 2.1.3. *We consider these asymptotic settings:*

- $T \to \infty$*, i.e. the total number of topics grows to infinity.*
- ***(Sparse documents):*** $\tau \to \infty, \tau = o(T)$*, i.e. the number of topics in each document also grows to infinity, but much smaller than the total number of topics. (This is a common parameter regime: we typically think of each document as a sparse combination of topics.)*
- ***(No sparsely supported topics):*** $v > (\frac{1}{1-(1-p_c)p_m} + 1)^2 + 1$ *(v is the number of tokens in each topic. $v \geq 10$ suffices under Assumption 2.1.4. This is also a common regime, where we assume no topic consists only of a small number of words.)*

Assumption 2.1.4. *In the training objective (Section 2.1.1), we consider the case* $p_m < \frac{1}{2},\ p_c = p_r \in (0, \frac{1}{2})$*.* [4]

Theorem 2.1.3 (Optimal attention weights)**.** *Suppose the data distribution follows the topic modeling assumption in Section 2.1.1 and Assumption 2.1.1. Suppose we train a single layer transformer given by* (2.7) *with* $\boldsymbol{b}^{pred} = 0$ *and* $\boldsymbol{W}^V$ *frozen to the optima in Theorem 2.1.2, under masked language modeling objective ((2.1)) with the squared loss ((2.3)), under Assumption 2.1.2, Assumption 2.1.3, and Assumption 2.1.4. Then, the optimal* (α, β) *satisfy*

$$\frac{v-1}{v}\alpha + \frac{1}{v}\beta \in (\lambda_1(\tau - 1), \lambda_2 T)$$

in which $\lambda_1 \coloneqq \frac{(1-(1-p_c)p_m + p_m p_r)(1+(1-p_c)p_m)}{2(1-(1-p_c)p_m)}$ *and* $\lambda_2 \coloneqq 100(\frac{1-(1-p_c)p_m}{p_m p_r} + 1)$*.*

[4] This setting is consistent with the masking scheme proposed in Devlin et al. (2019).

Remark 2.1.4. *In particular, Theorem 2.1.3 implies that if we choose τ, T such that the lower bound exceeds 1, we expect the attention between same-topic words to be* on average *larger than that between different-topic words.*

Remark 2.1.5. *Note that when $\boldsymbol{W}^V$ is block-diagonal with uniform blocks, it is impossible to meaningfully bound α or β individually; instead, only their weighted average $(\frac{v-1}{v}\alpha+\frac{1}{v}\beta)$ matters. In other words, different (α, β) will incur the same loss, as long as the above weighted average remains the same. Intuitively, this is because such block-diagonal $\boldsymbol{W}^V$ with uniform blocks* sums up *the attention on all words in each topic, and make predictions* solely based on the sums. *The proof of Theorem 2.1.3 is deferred to Section 2.1.7.7.*

Remark 2.1.6. *When there is no L_2-regularization, the first-stage optima of $\boldsymbol{W}^V$ is not unique. We include additional analysis for representative cases of $\boldsymbol{W}^V$ in Section 2.1.7.8.*

Remark 2.1.7. *When T, τ are finite, the loss expression turns out to be too complicated to characterize in closed form (because all the $o(1)$ terms need to be expanded). So we instead numerically compute the loss landscape as a function of α and β. See Section 2.1.5.5.*

2.1.5 Experiments

We analyze properties of the training dynamics via extensive experimental analysis. We will describe both the setup for synthetic (LDA-generated) data, and for Wikipedia data.

2.1.5.1 Results on synthetic LDA-generated data

Experimental setup In our experiments, we generate data following Section 2.1.1 with $T = 10, v = 10$, N uniformly randomly chosen from $[100, 150]$, except that Step 1 is changed to sampling the topic distribution according to the Dirichlet distribution (consistent with LDA, Blei et al., 2003) with $\alpha = 0.1$. Most sentences contain 2 to 4 topics. Our training objective follows Section 2.1.1 with $p_m = 0.15, p_c = 0.1, p_r = 0.1$ following Devlin et al. (2019). We use the model architecture following Section 2.1.1 but add back the bias terms $\boldsymbol{b}^K, \boldsymbol{b}^Q, \boldsymbol{b}^V$, following standard implementation in Wolf et al. (2020).

Trained token embeddings In Figure 2.1, we show that for a model in which all components are trained, the learned embedding weight $\boldsymbol{W}^E$ is such that $\boldsymbol{W}^{E^\top}\boldsymbol{W}^E$ displays a block-wise pattern. In particular, a diagonal pattern is a special case. These results show that our theory in Section 2.1.3 characterizes the optima of embedding layer which can be found by using either cross-entropy or squared losses, either SGD or Adam optimizers, and even when the other layers in the model are trained instead of frozen.

Learned value matrix W^V We show that when the word embeddings are *frozen to one-hot* and the attention weights are uniform (by setting $\boldsymbol{W}^K = 0, \boldsymbol{W}^Q = 0$), the trained $\boldsymbol{W}^V$ has a block-wise pattern, corresponding to the topical structure (see Figure 2.2).

We show (in Figure 2.6 in Section 2.1.5.2) that even when the attention weights $\boldsymbol{W}^K, \boldsymbol{W}^Q$ are jointly trained with $\boldsymbol{W}^V$, the model would still approximately converge to the type of block-wise $\boldsymbol{W}^V$ described in our analyses in Section 2.1.4.2.

Convergence point of trained attention weights We show that, our conclusion in Theorem 2.1.3 holds not just when $\boldsymbol{W}^V$ is *frozen* to a block-wise pattern, but also when it is *trained* and naturally converges to such pattern. And we show (in Table 2.1 in Section 2.1.5.3) that on average, each word pays more attention to words of *the same topic* than to words of *different topics*.

2.1.5.2 Additional results on learned value matrix $\boldsymbol{W}^V$

In Theorem 2.1.2 and Figure 2.2 we have shown that when freezing uniform attention weights and one-hot word embedding, under L_2-regularization, training a single layer transformer on our synthetic topic modeling distribution (Section 2.1.1) would make its $\boldsymbol{W}^V$ converge to a block-wise pattern that encodes the topic structure.

In the following Figure 2.5, we additionally show empirical results *without* L_2-regularization, matching our theory in Theorem 2.1.4.

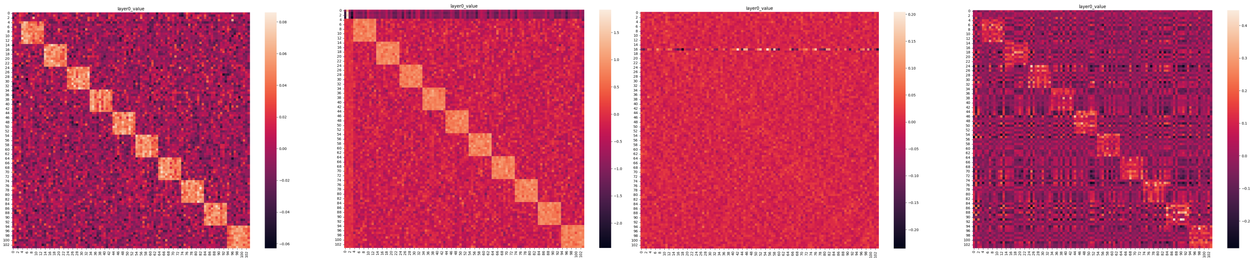


Figure 2.5: Convergence point of trained $\boldsymbol{W}^V$ (no L_2-regularization) when freezing uniform attention weights and one-hot word embedding. The four plots correspond to different combinations of loss function and optimizer. (Left to right) cross-entropy with SGD, cross-entropy with Adam, squared loss with SGD, squared loss with Adam, all using learning rate 0.01. The block-wise pattern verifies our theory in Section 2.1.4.2. The 10 blocks correspond to the 10 topics in the data distribution. In particular, in the third figure, the blocks are very weak and not easily visible, but we checked that the mean of the 1000 entries corresponding to the block positions is 0.00552563, which is over 10x the magnitude of the mean of a random subset of 1000 non-block entries (mean -0.00015675332, stdev 0.00060286524).

Complementing our experimental results in Section 2.1.5, Figure 2.6 shows that even when the attention weights $\boldsymbol{W}^K, \boldsymbol{W}^Q$ are *jointly trained* with $\boldsymbol{W}^V$, the model would still approximately converge to the type of block-wise $\boldsymbol{W}^V$ described in our analyses in Section 2.1.4.2.

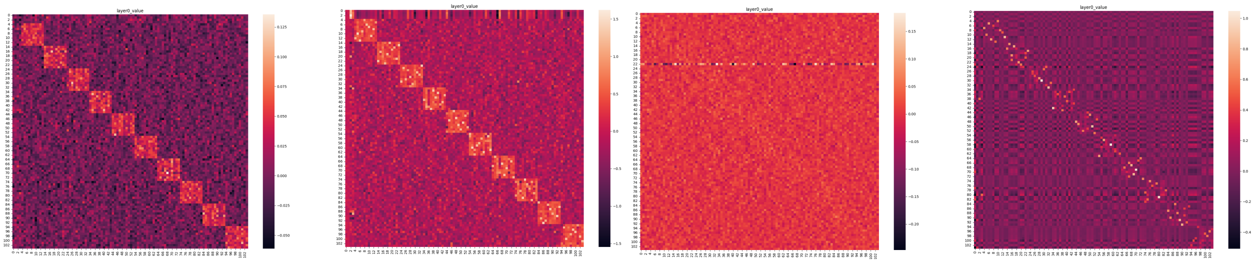


Figure 2.6: Convergence point of trained $\boldsymbol{W}^V$ when freezing one-hot word embedding but training attention weights. (Left to right) cross-entropy with SGD, cross-entropy with Adam, squared loss with SGD, squared loss with Adam, all using learning rate 0.01. The block-wise pattern shows that our analysis in Section 2.1.4.2 closely approximates the empirical training dynamics when $\boldsymbol{W}^K, \boldsymbol{W}^Q, \boldsymbol{W}^V$ are trained jointly. The 10 blocks correspond to the 10 topics in the data distribution. In particular, in the third figure, the blocks are very weak and not easily visible, but we checked that the mean of the 1000 entries corresponding to the block positions is 0.006545205, which is over 10x the magnitude of the mean of a random subset of 1000 non-block entries (mean -0.0006503917, stdev 0.0006370574).

2.1.5.3 Additional results on learned attention weights

Complementing our experimental results in Section 2.1.5, Table 2.1 shows that when the trained $\boldsymbol{W}^V$ is closer to **uniform within each block**, i.e. on average, each word pays more attention to *different words of the same topic* than to words of *different topics*.

Optimizer and Learning Rate	Avg Same-Word Attention	Avg Same-Topic- -Different-Word Attention	Avg Different-Topic Attention
Adam 0.003	0.00759 ± 0.00171	0.0108 ± 0.000657	0.00689 ± 0.000160
Adam 0.01	0.00811 ± 0.000705	0.010 ± 0.000392	0.00707 ± 0.000178
Adam 0.03	0.00453 ± 0.000346	0.0116 ± 0.000460	0.00665 ± 0.000200
SGD 0.01	0.0105	0.0106	0.00673
SGD 0.03	0.0140 ± 0.00158	0.0103 ± 0.000357	0.00641 ± 0.0000239

Table 2.1: Average attention weights when the model (with one-hot word embeddings) is trained under the cross-entropy loss and $\boldsymbol{W}^V$ converges to a block-wise pattern with closer to uniform blocks. We report mean ± std. deviation over 3 runs. The row "SGD 0.01" only contains 1 run, and the row "SGD 0.003" is removed, because these models had much higher final train and dev losses than others. For these failed runs, all three types of attention weights have similar averages, a sign that $\boldsymbol{W}^K$ and $\boldsymbol{W}^Q$ did not learn meaningful topical structures. Note that under most settings, *same-word* attention is larger than *same-topic-different-word* attention, which is larger than *different-topic* attention, verifying our conclusion in Theorem 2.1.3. The models trained using "Adam 0.03" has larger *same-topic-different-word* attention, which possibly made it unnecessary to rely on *same-word* attention to achieve a low loss, though our theory suggests that increasing *same-word* attention could further reduce the loss.

On the other hand, when the trained $\boldsymbol{W}^V$ is closer to **a diagonal pattern**, the above ordering is partially reversed, Table 2.2 shows that on average, each word pays the most attention to the *same word* in the document, followed by words of *different topics*, and the least attention to *different words of the same topic*.

Learning Rate	Avg Same-Word Attention	Avg Same-Topic- -Different-Word Attention	Avg Different-Topic Attention
0.003	0.0916 ± 0.000901	0.00185 ± 0.000170	0.00256 ± 0.0000332
0.01	0.0918 ± 0.00244	0.00182 ± 0.000474	0.00256 ± 0.000109

Table 2.2: Average attention weights when the model is trained under the cross-entropy loss with the Adam optimizer and $\boldsymbol{W}^V$ converges to a diagonal pattern. We report mean ± std. deviation over 7 runs, selected out of 10, by removing the runs in which the diagonal pattern in $\boldsymbol{W}^V$ is not visible or weak. Note that on average, *same-word* attention is larger than *different-topic* attention, which is larger than *same-topic-different-word* attention, verifying our conclusion in Theorem 2.1.5.

Model	Ambiguity Threshold	Avg embedding Cosine Similarity (Same-topic/Diff-topic)	Avg embedding Dot Product (Same-topic/Diff-topic)	Avg attn weight (Same-topic /Diff-topic)
Bert	0.0005	1.21	1.19	1.32
	0.001	1.13	1.15	1.28
	0.002	1.11	1.13	1.22
Albert	0.0005	5.64	6.29	1.33
	0.001	4.18	3.74	1.28
	0.002	3.24	2.93	1.22
Bart	0.0005	2.80	2.67	1.35
	0.001	1.95	1.92	1.31
	0.002	1.63	1.62	1.23
Electra	0.0005	5.98	5.37	2.14
	0.001	7.70	7.35	2.09
	0.002	7.46	8.08	1.95
Roberta	0.0005	6.44	6.81	1.40
	0.001	5.73	6.31	1.31
	0.002	5.24	5.30	1.22
Bert (randomly initialized)	0.0005	1.00080	1.00063	0.99943
	0.001	0.99974	1.00036	0.99996
	0.002	1.00016	1.00027	1.00007

Table 2.3: For models pretained on Wikipedia dataset, their token embeddings and attention weights encode topic structure. The different columns are: (1) The "ambiguity threshold", i.e. the number of words per topic, divided by the vocabulary size; **each word is only assigned one topic.** (2) The average embedding cosine similarity between different words of the *same topic*, divided by that between words of *different topics*. (3) The average embedding dot product between different words of the *same topic*, divided by that between words of *different topics*. (4) The average attention weight between different words of the *same topic*, divided by that between words of *different topics*. (The attention weights are normalized for debiasing, see discussion below for more details). Different rows represent different evaluation settings, controlled by "ambiguity threshold". Note that the avg same-topic embedding similarity and attention weight are consistently greater than the avg diff-topic counterparts, verifying our conclusions in Theorem 2.1.1 and Theorem 2.1.3.

2.1.5.4 Results on natural language data

For a set of pre-trained transformer-based models (and their corresponding tokenizers) downloaded from Huggingface (Wolf et al., 2020), we compare the embedding similarity and attention weights between same-topic tokens and different-topic tokens. The topics are determined by fitting an LDA model with 100 topics on a sample of Wikipedia corpus (WikimediaFoundation, 2023) tokenized by the above tokenizers. We filter stop words. For each topic, we only keep a fraction of tokens that LDA assigns the highest likelihood in this topic. Consistent with our theoretical setting, we restrict to keeping only one topic for each word. In Table 2.3, we provide the results after such pre-processing. We provide additional details about the experimental setup and additional results (including when the last restriction of "one topic per word" is removed) in Table 2.4.

In particular, for fair comparison, we should focus on the embedding similarity and attention weights between *different words of the same topic* and *different words of different topics*. (This is because those metrics are less meaningful for a pair of two *same words*, since their embeddings dot product is expected to be larger, which further biases the attention score comparisons.)

Ambiguity filter We also note that, for each word, an LDA model assigns some probability distribution of its topics. To determine whether two words are of the same topic, it is more meaningful if they share a topic in which both words have high likelihood. (By contrast, if two words each has some rarely-used topic that happens to overlap, we intuitively think of them as having different topics.)

To formalize such intuition, we filter out stop tokens, and other tokens that are not central to any topic (determined by the LDA). That is, for each topic t, LDA assigns to it a likelihood p_i for each word w_i in the vocabulary (of size n). We sort these (word, likelihood) pairs by decreasing likelihood:

$$(w_1, p_1), \cdots, (w_n, p_n)$$

then for a pre-defined threshold parameter $\theta \in (0, 1)$ controlling the proportion of words to be assigned to each topic, we only consider the topic t to contain the following words

$$\{w_i : i \leq \theta n\}$$

Debiasing average attention weight Moreover, we note that sentence length may cause a bias in attention weights calculation: intuitively, the average attention weight is the inverse of sentence length, but longer sentences usually contain more topics (and hence a larger proportion of different-topic word pairs). Thus, we expect that the average attention weight between different-topic word pairs are smaller than that between same-topic word pairs, *even for a transformer with random parameters.* (Empirically this bias indeed exists robustly, both on synthetic data and on Wikipedia data.) Therefore, we debias the effect of sentence length on attention weights: for each sentence, while computing the pairwise attention weights among its words, we "normalize the sentence length to 100", that is, we multiply the raw attention weights by sentence length, and then divide the result by 100. In this way, the average attention weight in each sentence is always $\frac{1}{100}$, regardless of the proportion of same-topic and different-topic word pairs. Indeed, as Table 2.3 and Table 2.4 show, for a randomly initialized BERT model, after our debiasing, the average same-topic and different-topic attention weights are roughly equal.

Results For a set of pre-trained transformer-based models downloaded from Huggingface (Wolf et al., 2020), we compare the embedding similarity and attention weights between same-topic tokens and different-topic tokens. The topics are determined by fitting an LDA model with 100 topics on a sample of tokenized Wikipedia corpus. We apply the above-mentioned ambiguity filter and debiasing.

- When we further restrict to keeping only one topic for each word (to be consistent with the setting in our theoretical analysis): see Table 2.3.
- Without the last restriction above: see the following Table 2.4.

Model	**Ambiguity Threshold**	**Avg embedding Cosine Similarity (Same-topic/Diff-topic)**	**Avg embedding Dot Product (Same-topic/Diff-topic)**	**Avg attn weight (Same-topic /Diff-topic)**
Bert	0.0005	1.14	1.04	1.23
	0.001	0.97	1.05	1.17
	0.002	0.99	0.93	1.13
Albert	0.0005	4.15	3.06	1.23
	0.001	3.09	3.04	1.17
	0.002	1.54	1.44	1.11
Bart	0.0005	2.51	1.76	1.27
	0.001	1.63	1.12	1.20
	0.002	1.06	0.85	1.11
Electra	0.0005	5.28	3.99	1.70
	0.001	5.56	5.57	1.58
	0.002	6.39	5.61	1.48
Roberta	0.0005	4.39	5.01	1.19
	0.001	5.20	4.25	1.15
	0.002	4.71	4.15	1.12
Bert (randomly initialized)	0.0005	0.99814	0.99957	1.00009
	0.001	0.99820	1.00167	1.00013
	0.002	0.99964	0.99928	0.99978

Table 2.4: For models pretained on Wikipedia dataset, their token embeddings and attention weights encode topic structure. The different columns are: (1) The "ambiguity threshold", i.e. the number of words per topic, divided by the vocabulary size; each word is only assigned one **or more** topic(s) (2) The average embedding cosine similarity between different words of the *same topic*, divided by that between words of *different topics*. (3) The average embedding dot product between different words of the *same topic*, divided by that between words of *different topics*. (4) The average attention weight between different words of the *same topic*, divided by that between words of *different topics*. (The attention weights are normalized for debiasing). Different rows represent different evaluation settings, controlled by "ambiguity threshold". Note that the avg same-topic embedding similarity and attention weight are mostly greater than the avg diff-topic counterparts (with some exceptions). Allowing multiple topics per word is different from our theoretical setup, so our conclusions in Theorem 2.1.1 and Theorem 2.1.3 do not cover this setting, though we conjecture that some variants of these theoretical results can be proven using similar approaches to ours.

2.1.5.5 Loss landscape with respect to attention weights in the non-asymptotic setting

When T, τ are finite, the loss expression turns out to be too complicated to characterize in closed form (because all the $o(1)$ terms need to be expanded). So we instead numerically compute the loss landscape as a function of α and β.

We set $T = 100$ following our experimental setup on Wikipedia dataset (in Section 2.1.5), and $v = 300$ (so total vocabulary size $Tv = 30000$) following the pre-trained BERT tokenizer in Huggingface implementation Wolf et al. (2020). We will vary $\tau \in \{20, 40, 60, 80\}$.

Diagonal $\boldsymbol{W}^V$ First, when $\boldsymbol{W}^V$ is fixed to a diagonal structure (Definition 2.1.2), Theorem 2.1.5 predicts that the loss is lowest when β is within an interval (boundaries controlled by τ and T), and α is less than a constant multiple of β. Both constraints are visible in the non-asymptotic setting, as we show in the following:

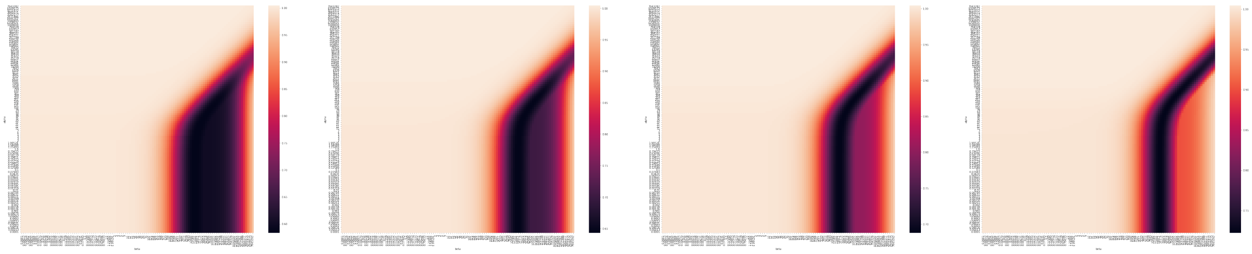


Figure 2.7: Landscape of squared loss under diagonal $\boldsymbol{W}^V$ (Definition 2.1.2), $T = 100, v = 300$. (left-to-right) $\tau = 20$, $\tau = 40$, $\tau = 60$, $\tau = 80$. In each plot, we perform a grid search over $\alpha, \beta \in [10^{-4}, 10^7]$ (both axes use log-scale). Darker color represents lower loss. Across a wide range of τ (compared to T), the loss is lowest when β is within an interval (lower bound growing with τ), and the optimal α is less than a constant multiple of β.

$\boldsymbol{W}^V$ with uniform blocks On the other hand, when $\boldsymbol{W}^V$ is fixed to a block-wise structure with uniform blocks (i.e. optima in Theorem 2.1.2), Theorem 2.1.3 predicts that the loss is lowest when a convex combination of α and β is within an interval (boundaries controlled by τ and T). As we show in the following, a variant of this constraint visibly holds in the non-asymptotic setting.

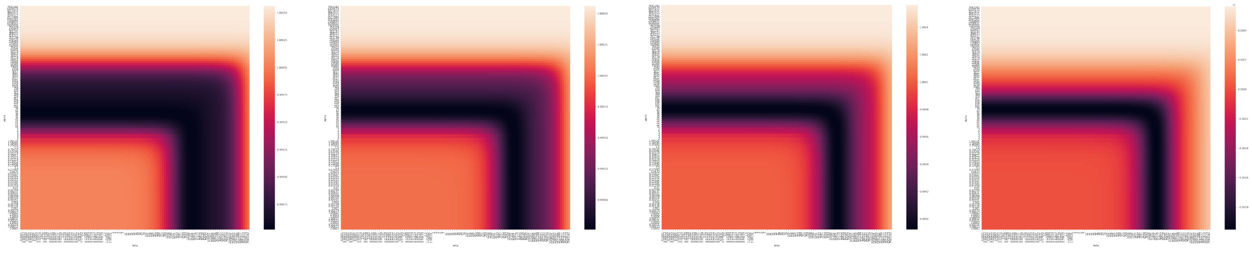


Figure 2.8: Landscape of squared loss for block-wise $\boldsymbol{W}^V$ with uniform blocks (i.e. optima in Theorem 2.1.2), $T = 100, v = 300$. (left-to-right) $\tau = 20$, $\tau = 40$, $\tau = 60$, $\tau = 80$. In each plot, we perform a grid search over $\alpha, \beta \in [10^{-4}, 10^7]$ (both axes use log-scale). Darker color represents lower loss. Across a wide range of τ (compared to T), the loss is lowest when (α, β) is in some corner-shaped region (both α and β are within some intervals whose lower bounds grow with τ).

2.1.6 Discussion

2.1.6.1 The two-stage optimization process

This two-stage optimization process (Section 2.1.4.1 and Figure 2.4) can be thought of as one iteration of the alternating optimization procedure. That is, we first train $\boldsymbol{W}^V$ while freezing $(\boldsymbol{W}^K, \boldsymbol{W}^Q)$, and then freeze $\boldsymbol{W}^V$ while training $(\boldsymbol{W}^K, \boldsymbol{W}^Q)$, and repeat this process.

In practice, $\boldsymbol{W}^K, \boldsymbol{W}^Q, \boldsymbol{W}^V$ in transformers are typically trained jointly instead of alternatingly. However, our empirical results show that, the conclusions drawn from the two-stage optimization analysis carry over even when they are trained jointly. Moreover, we don't find any qualitative aspects of normal training that are not captured by this two-stage approximation.

Intuitively, such two-stage phenomena occurs because if $\boldsymbol{W}^K, \boldsymbol{W}^Q, \boldsymbol{W}^V$ are initialized to random matrices near zero, and simultaneously trained, then in the initial steps, $\nabla_{\boldsymbol{W}^K} L$ contains the term $\boldsymbol{W}^Q$ (see (2.5)), which is close to 0. By contrast, $\nabla_{\boldsymbol{W}^V} L$ contains the softmax-normalized attention weights $A(\tilde{\boldsymbol{X}})$ (see (2.7)). Comparing these two, we shall see that $\nabla_{\boldsymbol{W}^V} L$ tends to be of larger in magnitude than $\nabla_{\boldsymbol{W}^K} L$, because each column of $\boldsymbol{W}^Q$ sums up to approximately 0, whereas each column of $A(\tilde{\boldsymbol{X}})$ sums up to exactly 1.

Therefore, in the initial steps (i.e. Stage 1), $\boldsymbol{W}^V$ intuitively grows much faster than $\boldsymbol{W}^K$. For the same reason (note the symmetry between $\boldsymbol{W}^K$ and $\boldsymbol{W}^Q$, see (2.5)), $\boldsymbol{W}^V$ intuitively grows much faster than $\boldsymbol{W}^Q$, too.

In Stage 2, it is less intuitively clear why $\|\boldsymbol{W}^V\|_F$ tends to plateau. Note that empirically, even when $\|\boldsymbol{W}^V\|_F$ plateaus, the $\boldsymbol{W}^V$ matrix itself still fluctuates with non-vanishing step-by-step changes. (That is, in each step, $\boldsymbol{W}^V$ "locally rotates" around the origin with an approximately constant norm.) Hence we refer to our Stage 2 analysis (which freezes $\boldsymbol{W}^V$ itself) as a simplification. However, the final empirical convergence point of $\boldsymbol{W}^V$ matches our theoretical analysis.

We show in Figure 2.9 that an approximate version of this multi-stage phenomenon can be observed on multi-layer transformers trained on Wikipedia as well.

Finally, this two-stage phenomenon is sensitive to hyperparameters like initialization and learning rate. In Figure 2.4, the The training process is not usually visibly two-stage using the common default hyperparameters. We leave it as an interesting future work to theoretically analyze the training dynamics when the two-stage phenomenon is not present.

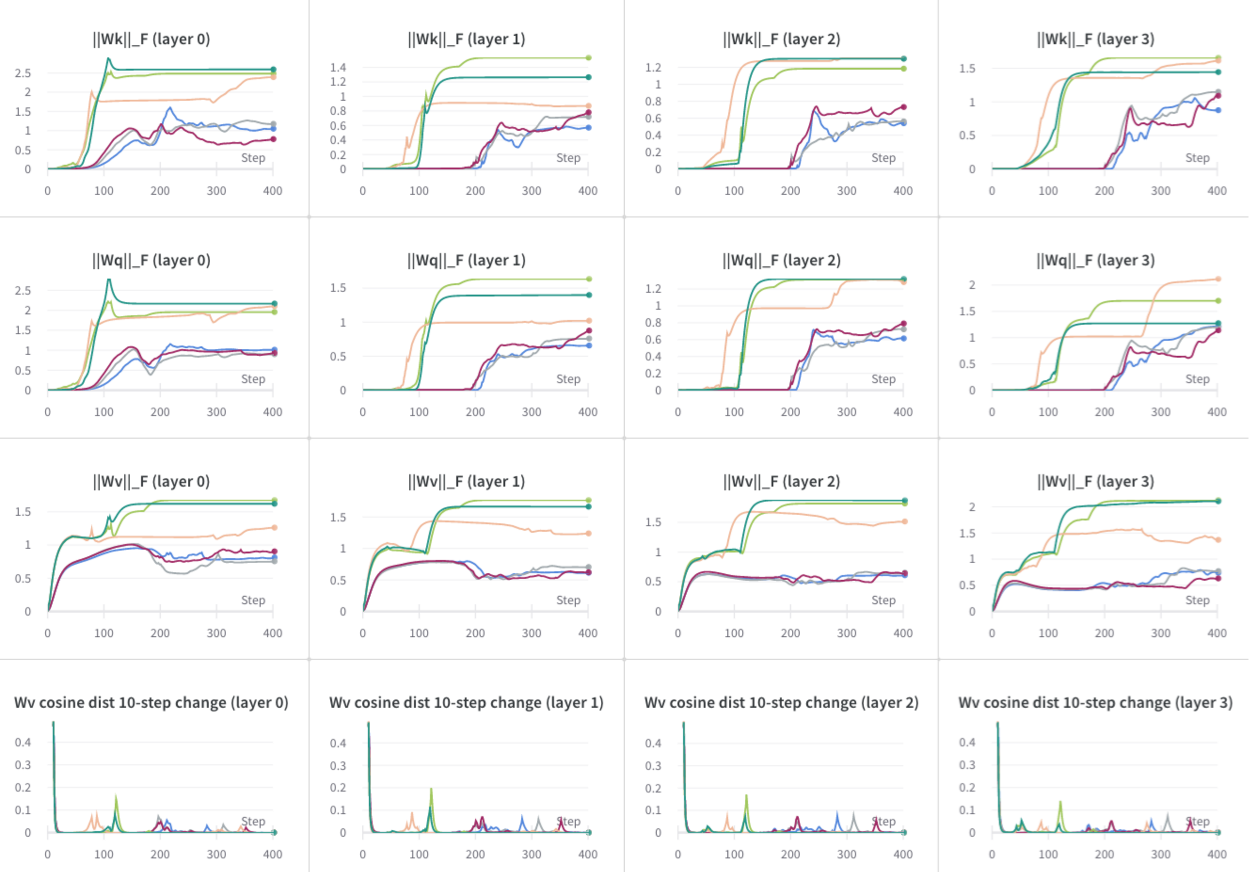


Figure 2.9: Two-stage learning dynamics of a 4-layer, 4-head-per-layer transformer trained on Wikipedia data. All weight matrices (key $\boldsymbol{W}^K$, query $\boldsymbol{W}^Q$, value $\boldsymbol{W}^V$ in each layer) are initialized to random matrices near zero, and *simultaneously trained*. Each column corresponds to one layer. The top 3 rows plot the trajectories of the Frobenius norms of $\boldsymbol{W}^K$, $\boldsymbol{W}^Q$, and $\boldsymbol{W}^V$ (weights from all heads in the same layer are concatenated together) after each gradient step. The bottom row measures the rotation of $\boldsymbol{W}^V$, i.e. the cosine distance between $\boldsymbol{W}^V$ in step t and $\boldsymbol{W}^V$ in step $(t-10)$. Cosine distance is defined as $\frac{1-cs}{2} \in [0,1]$, in which cs is the classic cosine similarity.

The initial 400 steps of the learning dynamics naturally exhibit an *approximately two-stage* phenomenon: in **Stage 1** (roughly steps 0-100), for all 4 layers, the norms of $\boldsymbol{W}^K$ and $\boldsymbol{W}^Q$ stay close to 0, while the norm of $\boldsymbol{W}^V$ increases significantly and the orientation of $\boldsymbol{W}^V$ changes rapidly. In **Stage 2** (roughly steps 100-400), the norms of $\boldsymbol{W}^K$'s and $\boldsymbol{W}^Q$'s start increasing significantly, much later than $\boldsymbol{W}^V$ matrices do. Different curves in the figure correspond to different settings of the hyperparameters as well as different runs in each setting.

2.1.6.2 Do topic-wise behaviors perfectly correlate with co-occurrence counts?

Additionally, we note that fitting a topic model is closely related to word co-occurrence statistics, which raises the following question: should those empirical phenomenon (i.e. higher same-topic attention and more similar same-topic embeddings, shown in Table 2.4) be more fundamentally attributed to larger co-occurrence counts?

In the following, we also compare them with some preliminary empirical results on the behavior of embedding and attention, from both topic modeling and co-occurrence perspectives. Specifically, we compare the average attention weights and average embedding dot products, between same-topic word pairs and the N pairs of words that co-occur the most frequently in a sample of the Wikipedia corpus. The cutoff N is determined so that the number of ”top co-occurring word pairs” is the same as the number of word pairs in each topic (controlled by the ambiguity threshold). The results are summarized in Table 2.5.

Based on those results, we conjecture that the topic-wise behavior of token embeddings and attention weights cannot be fully explained by simple co-occurrence counts.

Reasoning about their connections more formally would require analyzing some data distributions that better decouple these factors. We think that would be an interesting direction of future work.

# Word Pairs	Avg Attn Weight (Same-Topic)	Avg Attn Weight (Top Co-occur.)	Avg Embedding Cosine Similarity (Same-Topic)	Avg Embedding Cosine Similarity (Top Co-occur.)
105	0.00659	0.00751	0.468	0.316
435	0.00621	0.00695	0.461	0.311
1711	0.00597	0.00677	0.425	0.323

Table 2.5: For a BERT model pretained on Wikipedia dataset, the topic-wise behavior of its token embeddings and attention weights (shown in Table 2.3) cannot be fully explained by co-occurrence. The different columns are: (1) The number of pairs of tokens that have the highest co-occurrence counts (with stop tokens removed). The cutoffs are selected so that each row contains the same number of words pairs as one topic, corresponding to the rows in Table 2.3; (2) The average attention weights between same-topic words; (3) The average attention weights between tokens that co-occur the most; (4) The average embedding cosine similarity between different words of the *same topic*. (5) The average embedding cosine similarity between between tokens that co-occur the most. Note that for all “# word pairs” cutoffs considered, same-topic tokens have smaller average attention weight, but larger average embedding cosine similarity.

2.1.7 Proofs

2.1.7.1 Lemma 2.1.1 on the optimal linear transform when freezing uniform attention

Under our setting, we first prove the following useful Lemma 2.1.1. Intuitively, it states that, when freezing uniform attention, the output of self-attention weights essentially counts the *unmasked* tokens in the document (as a result of the masking process described in Section 2.1.1). Given those counts, the best way to predict a token at the masked positions in the *original* document (i.e. prior to the masking process) is to:

1. First, aggregate the counts of the unmasked words within each topic, to infer the topic distribution in the observed document. In this, we further have the restriction that:
   - Each unmasked word only contributes to predicting words of the *same topic*
   - Each unmasked word does not contribute to predicting words of *different topics*
   - Never predict the mask token (`[MASK]`), because the original document does not contain any `[MASK]`
2. Second, we "denoise" the topic distribution, i.e. we subtract the probability caused by filling in random words in the masking process (described in Section 2.1.1).

In line with our single layer transformer architecture (Section 2.1.1, (2.7)), we consider a special case in which the attention is *uniform*, i.e. $\forall i, j \in \{1, \cdots, N\}, A(\tilde{\boldsymbol{X}})_{ij} = \frac{1}{N}$, denoted by $A(\tilde{\boldsymbol{X}}) = \left[\frac{1}{N}\right]_{N\times N}$. (This can be achieved by setting $\boldsymbol{W}^K = 0, \boldsymbol{W}^Q = 0$.)

$$f(\tilde{\boldsymbol{X}}) = \boldsymbol{W}\tilde{\boldsymbol{X}}\left[\frac{1}{N}\right]_{N\times N} \tag{2.8}$$

which applies self-attention ((2.5)) on the one-hot representation of the masked document $\tilde{\boldsymbol{X}} \in \{0,1\}^{(Tv+1)\times N}$.

Lemma 2.1.1 (optimal linear transform when freezing uniform attention)**.** *Consider the simplified transformer architecture given by* (2.8) *with , as well as the masked language modeling objective (*(2.1)*) with squared loss (*(2.3)*). Then the set of minimizers* $\arg\min L(\boldsymbol{W})$ *consists of all* $\boldsymbol{W} \in \mathbb{R}^{(Tv+1)\times(Tv+1)}$ *that satisfy: there exist constants* $u_0, \cdots, u_{Tv} \in \mathbb{R}$ *such that*

1. *The 0-th row of* $\boldsymbol{W}$*:*
   (a) $\boldsymbol{W}_{00} = -\left(\frac{1}{p_m(1-p_c-p_r)} - 1\right) \cdot u_0$
   (b) $\forall t \in [T], \sum_{l\in t} \boldsymbol{W}_{0l} = u_0 v$
2. *The 0-th column of* $\boldsymbol{W}$*:*
   (a) $\forall i \in \{1, \cdots, Tv\}, \boldsymbol{W}_{i0} = -\frac{p_r}{(1-p_c-p_r)(1-(1-p_c)p_m)Tv} - \left(\frac{1}{(1-p_c-p_r)p_m} - 1\right) u_i$
3. $\boldsymbol{W}_{ij}$ *(*$\forall i, j \in \{1, \cdots, Tv\}$*):*
   (a) $\sum_{l\in \texttt{topic}(i)} \boldsymbol{W}_{il} = \frac{1}{1-(1-p_c)p_m} + u_i v$
   (b) $\forall t \in [T]$ *such that* $\texttt{topic}(i) \neq t$*,* $\sum_{l\in t} \boldsymbol{W}_{il} = u_i v$

Remark 2.1.8. *At the first glance, it might seem that the objective has a unique optima because it involves a squared loss, which is strongly convex. However, such uniqueness is undermined by the uniform attention condition:* $\boldsymbol{W}$ *is multiplied with a rank-1 matrix* $A(\tilde{\boldsymbol{X}}) = \left[\frac{1}{N}\right]_{N\times N}$*. This* $A(\tilde{\boldsymbol{X}})$ *will appear as a matrix multiplier in the Hessian of the objective with respect to* $\boldsymbol{W}$*, and so the Hessian is of rank 1, and therefore cannot have a positive minimum eigenvalue, implying that the objective is in fact* not *strongly convex.*

In fact, this optimization objective becomes strongly convex with an L_2 *regularization for some* $\lambda > 0$*.*

$$\arg\min_{\boldsymbol{W}^V} L_{MLM}(\boldsymbol{W}^V) + \lambda\|\boldsymbol{W}^V\|_F$$

Proof. For document $\boldsymbol{w}$ and the corresponding (masked) one-hot embedding $\tilde{\boldsymbol{X}}$:

$$\begin{aligned}
&\left[\tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}})\right]_{ij} \\
&= \frac{1}{N}\sum_{l=1}^{N} \tilde{\boldsymbol{X}}_{il} \quad \text{(i.e. independent of } j\text{)} \\
&= \frac{1}{N}\sum_{l=1}^{N} \mathbf{1}_{\tilde{\boldsymbol{X}}_{il}=1} \quad \text{(since } \tilde{\boldsymbol{X}} \text{ is one-hot)} \\
&= \begin{cases} p_m(1-p_c-p_r) & \text{if } i=0 \\ P_{\boldsymbol{w}}(i)(1-(1-p_c)p_m) + \frac{p_m p_r}{vT} & \text{if } i \in \{1,\cdots,Tv\} \end{cases} \quad \text{(by (2.17))}
\end{aligned}$$

Thus, the model prediction $\boldsymbol{W}\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}})$ satisfies

$$\begin{aligned}
(\boldsymbol{W}\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}}))_{ij} &= \boldsymbol{W}_{i0} p_m(1-p_c-p_r) + \sum_{l=1}^{Tv} \boldsymbol{W}_{il}\left(P_{\boldsymbol{w}}(l)(1-(1-p_c)p_m) + \frac{p_m p_r}{vT}\right) \\
&= \boldsymbol{W}_{i0} p_m(1-p_c-p_r) + (1-(1-p_c)p_m)\cdot \sum_{l=1}^{Tv} \boldsymbol{W}_{il} P_{\boldsymbol{w}}(l) + \frac{p_m p_r}{vT}\cdot \sum_{l=1}^{Tv} \boldsymbol{W}_{il} \\
&= \boldsymbol{W}_{i0} p_m(1-p_c-p_r) + (1-(1-p_c)p_m)\cdot \left(\sum_{l\in \texttt{topic}(i)} \boldsymbol{W}_{il} P_{\boldsymbol{w}}(i) + \sum_{l\notin \texttt{topic}(i)} \boldsymbol{W}_{il} P_{\boldsymbol{w}}(l)\right) + \frac{p_m p_r}{vT}\cdot \sum_{l=1}^{Tv} \boldsymbol{W}_{il}
\end{aligned} \tag{2.9}$$

and the last step follows since $\forall l \in \texttt{topic}(i), P_{\boldsymbol{w}}(l) = P_{\boldsymbol{w}}(i)$ under our setting in Section 2.1.1.

Recall that the loss is

$$L(\boldsymbol{W}) = \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}} \mathbb{E}_M \frac{1}{|M|}\sum_{j\in M} \|(\boldsymbol{W}\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}}))_{:j} - \boldsymbol{X}_{:j}\|_2^2$$

We will show that the average taken over $j \in M$ is the same as the average taken over all positions $j \in [N]$, by Assumption 2.1.1 and because M is uniformly randomly sampled from $[N]$. Moreover, note that $A(\tilde{\boldsymbol{X}}) = \left[\frac{1}{N}\right]_{N\times N}$, so $(\boldsymbol{W}\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}}))_{:j}$ is independent of j. The above observations imply that the loss can be simplified to

$$L(\boldsymbol{W}) = \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}} \frac{1}{N}\sum_{j=1}^{N} \|(\boldsymbol{W}\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}}))_{:j} - \boldsymbol{X}_{:j}\|_2^2$$

and so $L(\boldsymbol{W})$ is minimized when $\forall \boldsymbol{X}$,

$$(\boldsymbol{W}\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}}))_{:j} = \frac{1}{N}\sum_{l=1}^{N} \boldsymbol{X}_{:l}$$

which requires $\forall i \in \{0,\cdots,Tv+1\}$,

$$\begin{aligned}
(\boldsymbol{W}\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}}))_{0j} &= 0 \\
(\boldsymbol{W}\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}}))_{ij} &= P_{\boldsymbol{w}}(i), \quad \forall i \in \{1,\cdots,Tv\}
\end{aligned} \tag{2.10}$$

From (2.9) and (2.10) we get:

$$0 = \boldsymbol{W}_{00} p_m (1 - p_c - p_r) + (1 - (1 - p_c) p_m) \cdot \sum_{l=1}^{Tv} \boldsymbol{W}_{0l} P_{\boldsymbol{w}}(l) + \frac{p_m p_r}{vT} \cdot \sum_{l=1}^{Tv} \boldsymbol{W}_{0l}$$

$$P_{\boldsymbol{w}}(i) = \boldsymbol{W}_{i0} p_m (1 - p_c - p_r) + (1 - (1 - p_c) p_m) \cdot \left(\sum_{l \in \texttt{topic}(i)} \boldsymbol{W}_{il} P_{\boldsymbol{w}}(i) + \sum_{l \notin \texttt{topic}(i)} \boldsymbol{W}_{il} P_{\boldsymbol{w}}(l) \right) + \frac{p_m p_r}{vT} \cdot \sum_{l=1}^{Tv} \boldsymbol{W}_{il} \tag{2.11}$$

Note that under the topic modeling distribution in Section 2.1.1, for any topic $t \in [T]$,

$$P_{\boldsymbol{w}}((t-1)v + 1) = P_{\boldsymbol{w}}((t-1)v + 2) = \cdots P_{\boldsymbol{w}}(tv)$$

Hence we simplify (2.11) by considering the proportions of the "representative" tokens for each topic:

$$\{P_{\boldsymbol{w}}(tv) : t \in [T]\}$$

We obtain: for all sets of $\{P_{\boldsymbol{w}}(i) : i \in [Tv]\}$ satisfying our distribution in Section 2.1.1

$$0 = \boldsymbol{W}_{00} p_m (1 - p_c - p_r) + (1 - (1 - p_c) p_m) \cdot \sum_{t=1}^{T} \sum_{l \in t} \boldsymbol{W}_{0l} P_{\boldsymbol{w}}(tv) + \frac{p_m p_r}{vT} \cdot \sum_{t=1}^{T} \sum_{l \in t} \boldsymbol{W}_{0l} \tag{2.12}$$

and $\forall i \in \{1, \cdots, Tv\}$

$$P_{\boldsymbol{w}}(i) = \boldsymbol{W}_{i0} p_m (1 - p_c - p_r) + (1 - (1 - p_c) p_m) \cdot \left(\sum_{l \in \texttt{topic}(i)} \boldsymbol{W}_{il} P_{\boldsymbol{w}}(i) + \sum_{t \neq \texttt{topic}(i)} \sum_{l \in t} \boldsymbol{W}_{il} P_{\boldsymbol{w}}(tv) \right) + \frac{p_m p_r}{vT} \cdot \sum_{l=1}^{Tv} \boldsymbol{W}_{il} \tag{2.13}$$

Claim 2.1.1. *$\forall i \in \{1, \cdots, Tv\}, \exists u_i \in \mathbb{R}$ such that $\forall t \neq \texttt{topic}(i), \sum_{l \in t} \boldsymbol{W}_{il} = u_i v$. When $i = 0, \exists u_0 \in \mathbb{R}$ such that $\forall t \in [T], \sum_{l \in t} \boldsymbol{W}_{0l} = u_0 v$.*

Proof. $\forall i \in \{1, \cdots, Tv\}, \exists u_i \in \mathbb{R}$, suppose towards contradiction that $\exists t_1, t_2 \neq \texttt{topic}(i)$ such that $\sum_{l \in t_1} \boldsymbol{W}_{il} > \sum_{l \in t_2} \boldsymbol{W}_{il}$. We will show that (2.13) cannot hold for all sets of $\{P_{\boldsymbol{w}}(i) : i \in [Tv]\}$ satisfying our distribution in Section 2.1.1.

Specifically, fix $P_{\boldsymbol{w}}(i) = \frac{1}{2v}$ and consider the following settings of $\{P_{\boldsymbol{w}}(j) : j \notin \texttt{topic}(i)\}$:

- $P_{\boldsymbol{w}}(j) = \frac{1}{2v}$ if $\texttt{topic}(j) = t_1$ and 0 otherwise. Then (2.13) becomes

$$\frac{1}{2v} = \boldsymbol{W}_{i0} p_m (1 - p_c - p_r) + (1 - (1 - p_c) p_m) \cdot \left(\sum_{l \in \texttt{topic}(i)} \boldsymbol{W}_{il} \frac{1}{2v} + \sum_{l \in t_1} \boldsymbol{W}_{il} \frac{1}{2v} \right) + \frac{p_m p_r}{vT} \cdot \sum_{l=1}^{Tv} \boldsymbol{W}_{il}$$

- $P_{\boldsymbol{w}}(j) = \frac{1}{2v}$ if $\texttt{topic}(j) = t_2$ and 0 otherwise. Then (2.13) becomes

$$\frac{1}{2v} = \boldsymbol{W}_{i0} p_m (1 - p_c - p_r) + (1 - (1 - p_c) p_m) \cdot \left(\sum_{l \in \texttt{topic}(i)} \boldsymbol{W}_{il} \frac{1}{2v} + \sum_{l \in t_2} \boldsymbol{W}_{il} \frac{1}{2v} \right) + \frac{p_m p_r}{vT} \cdot \sum_{l=1}^{Tv} \boldsymbol{W}_{il}$$

Clearly the above two equations cannot both hold, because $\sum_{l \in t_1} \boldsymbol{W}_{il} > \sum_{l \in t_2} \boldsymbol{W}_{il}$.

Hence we proved by contradiction that $\forall t_1, t_2 \neq \texttt{topic}(i), \sum_{l \in t_1} \boldsymbol{W}_{il} = \sum_{l \in t_2} \boldsymbol{W}_{il}$. Likewise, when $i = 0$, $\forall t_1, t_2 in [T], \sum_{l \in t_1} \boldsymbol{W}_{0l} = \sum_{l \in t_2} \boldsymbol{W}_{0l}$.

□

By Claim 2.1.1, (2.12) becomes

$$\begin{aligned} 0 &= \boldsymbol{W}_{00}p_m(1-p_c-p_r) + (1-(1-p_c)p_m)\cdot\sum_{t=1}^{T} u_0 v P_{\boldsymbol{w}}(tv) + \frac{p_m p_r}{vT}\cdot\sum_{t=1}^{T} u_0 v \\ &= \boldsymbol{W}_{00}p_m(1-p_c-p_r) + (1-(1-p_c)p_m)\cdot u_0 + \frac{p_m p_r}{vT}\cdot T u_0 v \\ &= \boldsymbol{W}_{00}p_m(1-p_c-p_r) + (1-(1-p_c)p_m)\cdot u_0 + p_m p_r u_0 \\ &= \boldsymbol{W}_{00}p_m(1-p_c-p_r) + (1-(1-p_c-p_r)p_m)\cdot u_0 \end{aligned}$$

Therefore

$$\boldsymbol{W}_{00} = -\frac{(1-(1-p_c-p_r)p_m)\cdot u_0}{p_m(1-p_c-p_r)} = -\left(\frac{1}{p_m(1-p_c-p_r)} - 1\right)\cdot u_0$$

By Claim 2.1.1, (2.13) becomes

$$\begin{aligned} P_{\boldsymbol{w}}(i) &= \boldsymbol{W}_{i0}p_m(1-p_c-p_r) + (1-(1-p_c)p_m)\cdot\left(\sum_{l\in\texttt{topic}(i)} \boldsymbol{W}_{il}P_{\boldsymbol{w}}(i) + \sum_{t\neq\texttt{topic}(i)} u_i v P_{\boldsymbol{w}}(tv)\right) \\ &\quad + \frac{p_m p_r}{vT}\cdot(\sum_{l\in\texttt{topic}(i)} \boldsymbol{W}_{il} + (T-1)u_i v) \\ &= \boldsymbol{W}_{i0}p_m(1-p_c-p_r) + (1-(1-p_c)p_m)\cdot\left(\sum_{l\in\texttt{topic}(i)} \boldsymbol{W}_{il}P_{\boldsymbol{w}}(i) + u_i(1-vP_{\boldsymbol{w}}(i))\right) \\ &\quad + \frac{p_m p_r}{vT}\cdot(\sum_{l\in\texttt{topic}(i)} \boldsymbol{W}_{il} + (T-1)u_i v) \\ &= (1-(1-p_c)p_m)\,(\sum_{l\in\texttt{topic}(i)} \boldsymbol{W}_{il} - u_i v)P_{\boldsymbol{w}}(i) + \boldsymbol{W}_{i0}p_m(1-p_c-p_r) + (1-(1-p_c)p_m)\,u_i \\ &\quad + \frac{p_m p_r}{vT}\cdot(\sum_{l\in\texttt{topic}(i)} \boldsymbol{W}_{il} + (T-1)u_i v) \end{aligned}$$

Since this has to hold for all $P_{\boldsymbol{w}}(i) \in [0, \frac{1}{v}]$, the coefficients must match, i.e.

$$(1-(1-p_c)p_m)\,(\sum_{l\in\texttt{topic}(i)} \boldsymbol{W}_{il} - u_i v) = 1 \qquad (2.14)$$

$$\boldsymbol{W}_{i0}p_m(1-p_c-p_r) + (1-(1-p_c)p_m)\,u_i + \frac{p_m p_r}{vT}\cdot(\sum_{l\in\texttt{topic}(i)} \boldsymbol{W}_{il} + (T-1)u_i v) = 0 \qquad (2.15)$$

By (2.14),

$$\sum_{l\in\texttt{topic}(i)} \boldsymbol{W}_{il} = u_i v + \frac{1}{1-(1-p_c)p_m}$$

Plugging into (2.15),

$$\begin{aligned}
\boldsymbol{W}_{i0} &= -\frac{(1-(1-p_c)p_m)\,u_i + \frac{p_m p_r}{vT}\cdot(u_i v + \frac{1}{1-(1-p_c)p_m} + (T-1)u_i v)}{p_m(1-p_c-p_r)} \\
&= -\frac{(1-(1-p_c)p_m)\,u_i + \frac{p_m p_r}{vT}\cdot(\frac{1}{1-(1-p_c)p_m} + T u_i v)}{p_m(1-p_c-p_r)} \\
&= -\frac{(1-(1-p_c)p_m)\,u_i + \frac{p_m p_r}{vT(1-(1-p_c)p_m)} + p_m p_r u_i}{p_m(1-p_c-p_r)} \\
&= -\frac{p_r}{(1-p_c-p_r)vT(1-(1-p_c)p_m)} - \frac{(1-(1-p_c-p_r)p_m)}{p_m(1-p_c-p_r)}u_i \\
&= -\frac{p_r}{(1-p_c-p_r)(1-(1-p_c)p_m)Tv} - \left(\frac{1}{p_m(1-p_c-p_r)} - 1\right)u_i
\end{aligned}$$

□

2.1.7.2 Proof of Theorem 2.1.1: Optimal Token Embedding

Theorem (optimal token embedding, Theorem 2.1.1 restated)**.** *Consider training a transformer given by* (2.6) *with* $\boldsymbol{W}^K = 0, \boldsymbol{W}^Q = 0, \boldsymbol{W}^V = I$ *and* $\forall i \in \{1,\cdots,Tv\}, \boldsymbol{b}_i^{pred} = -\frac{p_m p_r}{(1-(1-p_c)p_m)Tv}$ *on data coming from the topic model described in Section 2.1.1, with the masked language modeling objective ((2.1)) with squared loss ((2.3)).*

Then, the optimal word embeddings $\boldsymbol{W}^E$ *are such that* $\boldsymbol{E} \coloneqq \boldsymbol{W}^{E^\top}\boldsymbol{W}^E$ *satisfies: there exist constants* $u_0,\cdots,u_{Tv} \in \mathbb{R}$ *such that*

1. *The 0-th row of* $\boldsymbol{E}$*:*
   (a) $\boldsymbol{E}_{00} = -\left(\frac{1}{p_m(1-p_c-p_r)} - 1\right)\cdot u_0$
   (b) $\forall t \in [T], \sum_{l\in t}\boldsymbol{E}_{0l} = u_0 v$
2. *The 0-th column of* $\boldsymbol{E}$*:*
   (a) $\forall i \in \{1,\cdots,Tv\}, \boldsymbol{E}_{i0} = -\left(\frac{1}{(1-p_c-p_r)p_m} - 1\right)u_i$
3. $\boldsymbol{E}_{ij}$ *(*$\forall i,j \in \{1,\cdots,Tv\}$*):*
   (a) $\sum_{l\in \texttt{topic}(i)}\boldsymbol{E}_{il} = \frac{1}{1-(1-p_c)p_m} + u_i v$
   (b) $\forall t \in [T]$ *such that* $\texttt{topic}(i) \neq t$*,* $\sum_{l\in t}\boldsymbol{E}_{il} = u_i v$

Proof. Under this setting, the model output is

$$\begin{aligned}
f(\tilde{\boldsymbol{X}}) &= \boldsymbol{W}^{E^\top}\boldsymbol{W}^E\tilde{\boldsymbol{X}}A(\boldsymbol{W}^E\tilde{\boldsymbol{X}}) + \boldsymbol{b}^{\text{pred}} \\
&= \boldsymbol{E}\tilde{\boldsymbol{X}}\frac{1}{N}\mathbf{1}_{N\times N} + \boldsymbol{b}^{\text{pred}} \\
&= \boldsymbol{E}'\tilde{\boldsymbol{X}}\frac{1}{N}\mathbf{1}_{N\times N}
\end{aligned} \tag{2.16}$$

in which $\mathbf{1}$ refers to the all-one matrix, and $\boldsymbol{E}' \in \mathbb{R}^{(Tv+1)\times(Tv+1)}$ is defined such that

$$\boldsymbol{E}'_{ij} = \begin{cases} \boldsymbol{E}_{ij} - \frac{p_r}{(1-p_r-p_c)(1-(1-p_c)p_m)Tv}, & \text{if } i \in \{1,\cdots,Tv\}, j = 0 \\ \boldsymbol{E}_{ij}, & \text{otherwise} \end{cases}$$

and the last step is because by (2.17),

$$\left(\tilde{\boldsymbol{X}}\frac{1}{N}\mathbf{1}_{N\times N}\right)_{0j} = p_m(1-p_c-p_r) \quad \forall j$$

and $\forall i \in \{1,\cdots,Tv\}$,

$$\begin{aligned}&\left(\boldsymbol{E}'\tilde{\boldsymbol{X}}\frac{1}{N}\mathbf{1}_{N\times N}\right)_{ij}\\ =&\left(\boldsymbol{E}\tilde{\boldsymbol{X}}\frac{1}{N}\mathbf{1}_{N\times N}\right)_{ij} - \frac{p_r}{(1-p_r-p_c)\left(1-(1-p_c)p_m\right)Tv}\cdot p_m(1-p_c-p_r)\\ =&\left(\boldsymbol{E}\tilde{\boldsymbol{X}}\frac{1}{N}\mathbf{1}_{N\times N}\right)_{ij} - \frac{p_m p_r}{\left(1-(1-p_c)p_m\right)Tv}\\ =&\left(\boldsymbol{E}\tilde{\boldsymbol{X}}\frac{1}{N}\mathbf{1}_{N\times N}\right)_{ij} + \boldsymbol{b}_i^{\text{pred}}\end{aligned}$$

Let $\boldsymbol{E}'^*$ denote any matrix in

$$\arg\min_{\boldsymbol{E}'} \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}}\mathbb{E}_M \frac{1}{|M|}\sum_{j\in M}\|(\boldsymbol{E}'\tilde{\boldsymbol{X}}\frac{1}{N}\mathbf{1}_{N\times N})_{:j} - \boldsymbol{X}_{:j}\|_2^2$$

then by Lemma 2.1.1, there exist constants $u_0,\cdots,u_{Tv}\in\mathbb{R}$ such that

1. The 0-th row of $\boldsymbol{E}'^*$:
   (a) $\boldsymbol{E}'^*_{00} = -\left(\frac{1}{p_m(1-p_c-p_r)}-1\right)\cdot u_0$
   (b) $\forall t\in[T], \sum_{l\in t}\boldsymbol{E}'^*_{0l} = u_0 v$
2. The 0-th column of $\boldsymbol{E}'^*$:
   (a) $\forall i\in\{1,\cdots,Tv\}, \boldsymbol{E}'^*_{i0} = -\frac{p_r}{(1-p_c-p_r)(1-(1-p_c)p_m)Tv} - \left(\frac{1}{(1-p_c-p_r)p_m}-1\right)u_i$
3. $\boldsymbol{E}'^*_{ij}$ $(\forall i,j\in\{1,\cdots,Tv\})$:
   (a) $\sum_{l\in\texttt{topic}(i)}\boldsymbol{E}'^*_{il} = \frac{1}{1-(1-p_c)p_m} + u_i v$
   (b) $\forall t\in[T]$ such that $\texttt{topic}(i)\neq t$, $\sum_{l\in t}\boldsymbol{E}'^*_{il} = u_i v$

Therefore, by (2.16), let $\boldsymbol{E}^*$ denote any matrix in

$$\arg\min_{\boldsymbol{E}} \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}}\mathbb{E}_M \frac{1}{|M|}\sum_{j\in M}\|(\boldsymbol{E}\tilde{\boldsymbol{X}}\frac{1}{N}\mathbf{1}_{N\times N})_{:j} + \boldsymbol{b}^{\text{pred}} - \boldsymbol{X}_{:j}\|_2^2$$

then there exist constants $u_0,\cdots,u_{Tv}\in\mathbb{R}$ such that

1. The 0-th row of $\boldsymbol{E}^*$:
   (a) $\boldsymbol{E}^*_{00} = -\left(\frac{1}{p_m(1-p_c-p_r)}-1\right)\cdot u_0$
   (b) $\forall t\in[T], \sum_{l\in t}\boldsymbol{E}^*_{0l} = u_0 v$
2. The 0-th column of $\boldsymbol{E}^*$:
   (a) $\forall i\in\{1,\cdots,Tv\}, \boldsymbol{E}^*_{i0} = -\left(\frac{1}{(1-p_c-p_r)p_m}-1\right)u_i$

3. $\boldsymbol{E}^*_{ij}$ ($\forall i, j \in \{1, \cdots, Tv\}$):
    (a) $\sum_{l \in \texttt{topic}(i)} \boldsymbol{E}^*_{il} = \frac{1}{1-(1-p_c)p_m} + u_i v$
    (b) $\forall t \in [T]$ such that $\texttt{topic}(i) \neq t$, $\sum_{l \in t} \boldsymbol{E}^*_{il} = u_i v$

Finally, note that a subset of this family of optima is *realizable*, in the sense that there exists such $\boldsymbol{E}^*$ and $u_0, \cdots, u_{Tv} \in \mathbb{R}$ s.t. there exists $\boldsymbol{W}^E \in \mathbb{R}^{d \times (Tv+1)}$ s.t. $\boldsymbol{E}^* = \boldsymbol{W}^{E^\top} \boldsymbol{W}^E$. The simplest example is

$$\begin{aligned} u_0, \cdots, u_{Tv} &= 0 \\ d &= Tv + 1 \\ \boldsymbol{E}^* &= \frac{1}{1-(1-p_c)p_m} I \\ \boldsymbol{W}^E &= \frac{1}{\sqrt{1-(1-p_c)p_m}} I \end{aligned}$$

□

2.1.7.3 Optimal $\boldsymbol{W}^V$ when freezing uniform attention without regularization

Theorem 2.1.4 (optimal $\boldsymbol{W}^V$ when freezing uniform attention)**.** *On the topic modeling data distribution described in Section 2.1.1, with the* $\texttt{topic}$ *relation defined in Definition 2.1.1, under Assumption 2.1.1, with a single layer transformer given by* (2.7) *whose* $\boldsymbol{W}^K = 0, \boldsymbol{W}^Q = 0, \boldsymbol{b}^{pred} = 0$*, under masked language modeling objective ((2.1)) with the squared loss ((2.3)),* $\arg\min L(\boldsymbol{W}^V)$ *consists of all* $\boldsymbol{W}^V \in \mathbb{R}^{(Tv+1)\times(Tv+1)}$ *that satisfy: there exist constants* $u_0, \cdots, u_{Tv} \in \mathbb{R}$ *such that*

1. *The 0-th row of* $\boldsymbol{W}^V$*:*
    (a) $\boldsymbol{W}^V_{00} = -\left(\frac{1}{p_m(1-p_c-p_r)} - 1\right) \cdot u_0$
    (b) $\forall t \in [T], \sum_{l \in t} \boldsymbol{W}^V_{0l} = u_0 v$
2. *The 0-th column of* $\boldsymbol{W}^V$*:*
    (a) $\forall i \in \{1, \cdots, Tv\}, \boldsymbol{W}^V_{i0} = -\frac{p_r}{(1-p_c-p_r)(1-(1-p_c)p_m)Tv} - \left(\frac{1}{(1-p_c-p_r)p_m} - 1\right) u_i$
3. $\boldsymbol{W}^V_{ij}$ *(*$\forall i, j \in \{1, \cdots, Tv\}$*):*
    (a) $\sum_{l \in \textit{topic}(i)} \boldsymbol{W}^V_{il} = \frac{1}{1-(1-p_c)p_m} + u_i v$
    (b) $\forall t \in [T]$ *such that* $\textit{topic}(i) \neq t$*,* $\sum_{l \in t} \boldsymbol{W}^V_{il} = u_i v$

Proof. Note that this is exactly the statement of Lemma 2.1.1 (proved in Section 2.1.7.1) in the case of $\boldsymbol{W} \coloneqq \boldsymbol{W}^V$. □

2.1.7.4 Proof of Theorem 2.1.2: case when adding L_2 regularization

Theorem (optimal $\boldsymbol{W}^V$ with mild L_2-regularization when freezing uniform attention, restated)**.** *On the topic modeling data distribution described in Section 2.1.1, with the* $\texttt{topic}$ *relation defined in Definition 2.1.1, under Assumption 2.1.1, with a single layer transformer given by* (2.7) *whose* $\boldsymbol{W}^K = 0, \boldsymbol{W}^Q = 0, \boldsymbol{b}^{pred} = 0$*, under the* L_2*-regularized masked language modeling objective* (2.2) *with the squared loss* (2.3)*,*

$$\lim_{\lambda \to 0} \arg\min L_{l2reg}(\boldsymbol{W}^V) = \{\boldsymbol{W}^{V*}\}$$

in which $\boldsymbol{W}^{V*} \in \mathbb{R}^{(Tv+1)\times(Tv+1)}$ *satisfies:*

1. *The 0-th row of* $\boldsymbol{W}^{V*}$*:*

   (a) $\forall j \in \{0, \cdots, Tv\}, \boldsymbol{W}^{V*}_{0j} = 0$

2. *The 0-th column of* $\boldsymbol{W}^{V*}$*:*

   (a) $\forall i \in \{1, \cdots, Tv\}, \boldsymbol{W}^{V*}_{i0} = \frac{c_2 c_3 - c_1 Tv}{c_2^2 + Tv}$

3. $\boldsymbol{W}^{V*}_{ij}$ *(*$\forall i, j \in \{1, \cdots, Tv\}$*):*

   (a) $\forall l \notin \texttt{topic}(i),\ \boldsymbol{W}^{V*}_{il} = \boldsymbol{W}^{V*}_{\textit{diff-topic}} \coloneqq -\frac{c_1 c_2 + c_3}{c_2^2 + Tv}$

   (b) $\forall l \in \texttt{topic}(i),\ \boldsymbol{W}^{V*}_{il} = \boldsymbol{W}^{V*}_{\textit{same-topic}} \coloneqq \boldsymbol{W}^{V*}_{\textit{diff-topic}} + \frac{c_3}{v}$

in which the constants

- $c_1 = \frac{p_r}{(1-p_c-p_r)(1-(1-p_c)p_m)Tv} \in (0, 1)$
- $c_2 = \frac{1}{(1-p_c-p_r)p_m} - 1 \in (0, +\infty)$
- $c_3 = \frac{1}{1-(1-p_c)p_m} \in (1, +\infty)$

Proof. We proceed in the following two steps.

Step 1: the optima converges to one outlined in Lemma 2.1.1

Let S denote the set of optima outlined in Lemma 2.1.1. Suppose towards contradiction that $\exists \boldsymbol{W}^{V*} \notin S$ such that $\boldsymbol{W}^{V*} \in \lim_{\lambda \to 0} \arg\min L_{\text{l2reg}}(\boldsymbol{W}^V)$.

In comparison, $\forall \boldsymbol{W} \in S$, by Lemma 2.1.1, since $\boldsymbol{W}^{V*} \notin S$,

$$L(\boldsymbol{W}) < L(\boldsymbol{W}^{V*})$$

Moreover, note that since $\|\boldsymbol{W}\|_F$ is finite,

$$\lim_{\lambda \to 0} \lambda \|\boldsymbol{W}\|_F^2 = 0 \leq \lim_{\lambda \to 0} \lambda \|\boldsymbol{W}^{V*}\|_F^2$$

Combining the above two observations gives

$$\lim_{\lambda \to 0} L_{\text{l2reg}}(\boldsymbol{W}) = L(\boldsymbol{W}) + \lim_{\lambda \to 0} \lambda \|\boldsymbol{W}\|_F^2 < L(\boldsymbol{W}^{V*}) + \lim_{\lambda \to 0} \lambda \|\boldsymbol{W}^{V*}\|_F^2 = \lim_{\lambda \to 0} L_{\text{l2reg}}(\boldsymbol{W}^{V*})$$

which contradicts $\boldsymbol{W}^{V*} \in \lim_{\lambda \to 0} \arg\min L_{\text{l2reg}}(\boldsymbol{W}^V)$.

Therefore, we have proved by contradiction that

$$\forall \boldsymbol{W}^V \in \lim_{\lambda \to 0} \arg\min L_{\text{l2reg}}(\boldsymbol{W}^V), \quad \boldsymbol{W}^V \in S$$

Step 2: solve for the coefficients that minimize the L_2 penalty

By Step 1,

$$\begin{aligned}
\lim_{\lambda \to 0} \arg\min L_{\text{l2reg}}(\boldsymbol{W}^V) &= \lim_{\lambda \to 0} \arg\min_{\boldsymbol{W}^V \in S} L_{\text{l2reg}}(\boldsymbol{W}^V) \\
&= \lim_{\lambda \to 0} \arg\min_{\boldsymbol{W}^V \in S} L(\boldsymbol{W}^V) + \lambda \|\boldsymbol{W}^V\|_F^2 \\
&= \lim_{\lambda \to 0} \arg\min_{\boldsymbol{W}^V \in S} \min L(\boldsymbol{W}^V) + \lambda \|\boldsymbol{W}^V\|_F^2 \\
&= \lim_{\lambda \to 0} \arg\min_{\boldsymbol{W}^V \in S} \lambda \|\boldsymbol{W}^V\|_F^2
\end{aligned}$$

in which the last step is because $\forall \boldsymbol{W}^V \in S$, $L(\boldsymbol{W}^V) = \min L(\boldsymbol{W}^V)$, which is a constant independent of $\boldsymbol{W}^V$.

Then it suffices to find the constants $u_0, \cdots, u_{Tv} \in \mathbb{R}$ that minimizes $\|\boldsymbol{W}^V\|_F$.

□

2.1.7.5 Helping lemmas on masking probabilities

In this section, we will calculate a few expressions for the masking probabilities, which will be useful for the proofs later on. We will also introduce a few constants for brevity of notation.

A straightforward calculation shows that the probabilities after the masking process satisfy:

Proposition 2.1.1 (Probabilities after masking)**.** *After the masking process as in Section 2.1.1 is applied to a document $\boldsymbol{w}$, the distribution for the new document $\tilde{\boldsymbol{w}}$ satisfies*

$$P_{\tilde{\boldsymbol{w}}}(i) = \begin{cases} \frac{1}{v\tau}(1-(1-p_c)p_m) + \frac{p_m p_r}{vT}, & \text{if } \texttt{topic}(i) \in \{t_1, \cdots, t_\tau\} \\ p_m(1-p_c-p_r), & \text{if } i = \texttt{[MASK]} := 0 \\ \frac{p_m p_r}{vT}, & \text{otherwise} \end{cases} \tag{2.17}$$

For convenience, we will introduce the notation

$$p_1 := \frac{1}{v\tau}(1-(1-p_c)p_m) + \frac{p_m p_r}{vT} \tag{2.18}$$

$$p_2 := \frac{p_m p_r}{vT} \tag{2.19}$$

Another straightforward calculation can be used to express the relationship between the constant c_3 in Assumption 2.1.2 and the α, β. Namely, we have:

Proposition 2.1.2 (Expressing c_3 in terms of α, β)**.** *The constant c_3 in Assumption 2.1.2 satisfies:*

$$c_3 = \begin{cases} \frac{1}{(\beta p_1 + \alpha p_1(v-1) + p_1 v(\tau-1) + p_2 v(T-\tau))N}, & \text{if } \tilde{w}_j \in \{t_i\}_{i\in[\tau]} \\ \frac{1}{(\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1))N}, & \text{if } \tilde{w}_j \in [T]\backslash\{t_i\}_{i\in[\tau]} \end{cases}$$

Again, for notational convenience, we will introduce z_1, z_2, s.t.

$$z_1 := \beta p_1 + \alpha p_1(v-1) + p_1 v(\tau-1) + p_2 v(T-\tau)$$
$$z_2 := \beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1)$$

Proof of Proposition 2.1.2. We will get these equalities by considering the marginalization constraints, depending on the topic of $\tilde{w}_j$. Consider first a j, such that $\tilde{w}_j \in \{t_i\}_{i\in[\tau]}$:

- Note, for every position i, with probability p_1, we have $\tilde{w}_i = \tilde{w}_j$, so $A(\tilde{\boldsymbol{X}})_{ij} = \beta c_3$ by Assumption 2.1.2.
- Note also, for every position i, with probabilitiy $p_1(v-1)$, we have $\tilde{w}_i \neq \tilde{w}_j$ but $\texttt{topic}(\tilde{w}_i) = \texttt{topic}(\tilde{w}_j)$, and so $A(\tilde{\boldsymbol{X}})_{ij} = \alpha c_3$.
- Finally, note that for every position i, with probability $(p_1 v(\tau-1) + p_2 v(T-\tau))$ we have $\texttt{topic}(\tilde{w}_i) \neq \texttt{topic}(\tilde{w}_j)$, so $A(\tilde{\boldsymbol{X}})_{ij} = c_3$.

Since $\sum_{i=1}^N A(\tilde{\boldsymbol{X}})_{ij} = 1$, we obtain $c_3 = \frac{1}{(\beta p_1 + \alpha p_1(v-1) + p_1 v(\tau-1) + p_2 v(T-\tau))N}$.

Consider next a j, s.t. $\tilde{w}_j \in [T]\backslash\{t_i\}_{i\in[\tau]}$. By similar considerations as before,

- With probability p_2, a position i in $\tilde{\boldsymbol{w}}$ satisfies $\tilde{w}_i = \tilde{w}_j$, so $A(\tilde{\boldsymbol{X}})_{ij} = \beta c_3$.
- With probability $p_2(v-1)$, a position i satisfies $\tilde{w}_i \neq \tilde{w}_j$ but $\texttt{topic}(\tilde{w}_i) = \texttt{topic}(\tilde{w}_j)$, so $A(\tilde{\boldsymbol{X}})_{ij} = \alpha c_3$.
- Finally, with probability $p_1 v\tau + p_2 v(T-\tau-1)$, a position i in satisfies $\texttt{topic}(\tilde{w}_i) \neq \texttt{topic}(\tilde{w}_j)$, so $A(\tilde{\boldsymbol{X}})_{ij} = c_3$.

Since $\sum_{i=1}^N A(\tilde{\boldsymbol{X}})_{ij} = 1$, we obtain $c_3 = \frac{1}{(\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1))N}$. The proposition thus follows. □

2.1.7.6 Implication of topic-wise attention assumption on model output

In this section we calculate the part $\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}})$ using the results of Section 2.1.7.5:

Proposition 2.1.3. *Using the calculation and notations of z_1 and z_2 in Section 2.1.7.5:*

$$
\begin{aligned}
&\left[\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}})\right]_{ij} = \sum_{l=1}^{N} \tilde{\boldsymbol{X}}_{il}A(\tilde{\boldsymbol{X}})_{lj} = \sum_{l=1}^{N} \mathbf{1}_{\tilde{\boldsymbol{X}}_{il}=1}A(\tilde{\boldsymbol{X}})_{lj} = P(\tilde{\boldsymbol{X}}_{il}=1)\cdot N \cdot A(\tilde{\boldsymbol{X}})_{lj} \\
&= \begin{cases}
p_1 N \frac{\beta}{z_1 N} = \frac{p_1\beta}{z_1}, & \textit{if } i = j,\ \boldsymbol{topic}(j) \in \{t_1, \cdots, t_\tau\} \textbf{\textit{(Same token)}} \\
p_1 N \frac{\alpha}{z_1 N} = \frac{p_1\alpha}{z_1}, & \textit{if } i \neq j,\ \boldsymbol{topic}(i) = \boldsymbol{topic}(j) \in \{t_1, \cdots, t_\tau\} \textbf{\textit{(Different token, same topic)}} \\
p_1 N \frac{1}{z_1 N} = \frac{p_1}{z_1}, & \textit{if } \boldsymbol{topic}(i) \neq \boldsymbol{topic}(j),\ \boldsymbol{topic}(i) \in \{t_1, \cdots, t_\tau\},\ \boldsymbol{topic}(j) \in \{t_1, \cdots, t_\tau\} \\
p_2 N \frac{1}{z_1 N} = \frac{p_2}{z_1}, & \textit{if } \boldsymbol{topic}(i) \neq \boldsymbol{topic}(j),\ \boldsymbol{topic}(i) \notin \{t_1, \cdots, t_\tau\},\ \boldsymbol{topic}(j) \in \{t_1, \cdots, t_\tau\} \\
p_2 N \frac{\beta}{z_2 N} = \frac{p_2\beta}{z_2}, & \textit{if } i = j,\ \boldsymbol{topic}(j) \notin \{t_1, \cdots, t_\tau\} \\
p_2 N \frac{\alpha}{z_2 N} = \frac{p_2\alpha}{z_2}, & \textit{if } i \neq j,\ \boldsymbol{topic}(i) = \boldsymbol{topic}(j) \notin \{t_1, \cdots, t_\tau\} \\
p_1 N \frac{1}{z_2 N} = \frac{p_1}{z_2}, & \textit{if } \boldsymbol{topic}(i) \neq \boldsymbol{topic}(j),\ \boldsymbol{topic}(i) \in \{t_1, \cdots, t_\tau\},\ \boldsymbol{topic}(j) \notin \{t_1, \cdots, t_\tau\} \\
p_2 N \frac{1}{z_2 N} = \frac{p_2}{z_2}, & \textit{if } \boldsymbol{topic}(i) \neq \boldsymbol{topic}(j),\ \boldsymbol{topic}(i) \notin \{t_1, \cdots, t_\tau\},\ \boldsymbol{topic}(j) \notin \{t_1, \cdots, t_\tau\}
\end{cases}
\end{aligned}
\tag{2.20}
$$

2.1.7.7 Proof of Theorem 2.1.3 (optimal attention when freezing $\boldsymbol{W}^V$ to uniform blocks)

Theorem (optimal attention weights when freezing block-wise $\boldsymbol{W}^V$, Theorem 2.1.3 restated)**.** *Suppose the data distribution follows the topic modeling assumption in Section 2.1.1 and Assumption 2.1.1. Suppose we train a single layer transformer given by (2.7) with $\boldsymbol{b}^{pred} = 0$ and $\boldsymbol{W}^V$ frozen to the optima in Theorem 2.1.2, under masked language modeling objective ((2.1)) with the squared loss ((2.3)), under Assumption 2.1.2, Assumption 2.1.3, and Assumption 2.1.4. Then, the optimal (α, β) satisfy*

$$\frac{v-1}{v}\alpha + \frac{1}{v}\beta \in (\lambda_1(\tau - 1), \lambda_2 T)$$

in which the constants $\lambda_1 := \frac{(1-(1-p_c)p_m + p_m p_r)(1+(1-p_c)p_m)}{2(1-(1-p_c)p_m)}$ *and* $\lambda_2 := 100(\frac{1-(1-p_c)p_m}{p_m p_r} + 1)$.

Proof. Define $\gamma := \frac{v-1}{v}\alpha + \frac{1}{v}\beta$.

Recall the architecture under consideration, i.e.

$$\hat{\boldsymbol{X}} := \boldsymbol{W}^V \tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}})$$

The squared loss ((2.3)) is

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}} \mathbb{E}_M \left[\frac{1}{|M|} \sum_{j \in M} l(f(\tilde{\boldsymbol{X}})_{:j}, \boldsymbol{X}_{:j}) \right]_{ij} \\
&= \frac{1}{p_m N} \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}} \mathbb{E}_M \left[\sum_{j:\tilde{w}_j = \texttt{[MASK]}} l(f(\tilde{\boldsymbol{X}})_{:j}, \boldsymbol{X}_{:j}) + \sum_{j \in M, \tilde{w}_j \neq \texttt{[MASK]}} l(f(\tilde{\boldsymbol{X}})_{:j}, \boldsymbol{X}_{:j}) \right] \\
&= \frac{1}{p_m N} \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}} \mathbb{E}_M \left[\sum_{j:\tilde{w}_j = \texttt{[MASK]}} \|(\boldsymbol{W}^V \tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}}))_{:j} - \boldsymbol{X}_{:j}\|_2^2 + \sum_{j \in M, \tilde{w}_j \neq \texttt{[MASK]}} \|(\boldsymbol{W}^V \tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}}))_{:j} - \boldsymbol{X}_{:j}\|_2^2 \right] \\
&= \frac{1}{p_m N} \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}} \mathbb{E}_M \left[\sum_{j:\tilde{w}_j = \texttt{[MASK]}} \|\boldsymbol{W}^V \tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}})_{:j} - \boldsymbol{X}_{:j}\|_2^2 + \sum_{j \in M, \tilde{w}_j \neq \texttt{[MASK]}} \|\boldsymbol{W}^V \tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}})_{:j} - \boldsymbol{X}_{:j}\|_2^2 \right]
\end{aligned}$$

Note that when $\tilde{w}_j = \texttt{[MASK]}$, $A(\tilde{\boldsymbol{X}})_{:j}$ is the attention from `[MASK]` to other tokens, and therefore is independent of the setting of α and β in Assumption 2.1.2. Thus, in the following, we only consider the case in which $j \in M, \tilde{w}_j \neq \texttt{[MASK]}$, namely, w_j is masked, but $\tilde{w}_j$ is chosen to be either the correct token or the random token. Hence define:

$$L(\gamma) := \frac{1}{p_m N} \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}} \mathbb{E}_M \left[\sum_{j \in M, \tilde{w}_j \neq \texttt{[MASK]}} \|\boldsymbol{W}^V \tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}})_{:j} - \boldsymbol{X}_{:j}\|_2^2 \right] \tag{2.21}$$

Note that $\forall \boldsymbol{y} \in \mathbb{R}^{Tv+1}$

$$(\boldsymbol{W}^V \boldsymbol{y})_i = \begin{cases} 0, & i = 0 \\ q(\frac{1}{v} \sum_{l \in \text{topic}(i)} y_l), & i \in \{1, \cdots, Tv+1\} \end{cases}$$

in which

$$q(x) := \frac{1}{1-(1-p_c)p_m} x - \frac{p_m p_r}{(1-(1-p_c)p_m)Tv}$$

In our context, we will consider $\boldsymbol{y} = \tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}})_{:j}$ in $L(\gamma)$ above.

For a document $\boldsymbol{w}$ which contains topics $t_1, \cdots, t_\tau \in [T]$, there are the following cases:

Case 1: $\texttt{topic}(\tilde{w}_j) = \texttt{topic}(w_j)$ When $\tilde{w}_j$ after masking belongs to the same topic as the correct token w_j. (This happens with probability $p_c + \frac{p_r}{T}$)

By (2.20),

$$\frac{1}{v}\sum_{l\in\text{topic}(i)}\left[\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}})\right]_{lj} = \begin{cases} \frac{1}{v}\left(\frac{p_1\beta}{z_1} + \sum_{l\in\text{topic}(i), l\neq\tilde{w}_j}\frac{p_1\alpha}{z_1}\right) = \frac{1}{v}\left(\frac{p_1\beta+(v-1)p_1\alpha}{z_1}\right) = \frac{p_1\gamma}{z_1}, & \text{if } \texttt{topic}(i) = \texttt{topic}(\tilde{w}_j) \\ \frac{1}{v}\left(\sum_{l\in\text{topic}(i)}\frac{p_1}{z_1}\right) = \frac{p_1}{z_1}, & \text{if } \texttt{topic}(i) \neq \texttt{topic}(\tilde{w}_j), \texttt{topic}(i) \in \{t_1, \cdots, t_\tau\} \\ \frac{1}{v}\left(\sum_{l\in\text{topic}(i)}\frac{p_2}{z_1}\right) = \frac{p_2}{z_1}, & \text{if } \texttt{topic}(i) \neq \texttt{topic}(\tilde{w}_j), \texttt{topic}(i) \notin \{t_1, \cdots, t_\tau\} \end{cases} \tag{2.22}$$

Recall that the label is

$$\boldsymbol{X}_{:j} = \begin{cases} 1, & i = w_j \\ 0, & i \in \{0, \cdots, Tv\}\backslash w_j \end{cases}$$

Hence the contribution to the loss from token $\tilde{w}_j$ is

$$\begin{aligned} &(p_c + \frac{p_r}{T})[\left(1 - q\left(\frac{p_1\gamma}{z_1}\right)\right)^2 + q(\frac{p_1\gamma}{z_1})^2\cdot(v-1) + q(\frac{p_1}{z_1})^2\cdot v(\tau-1) + q(\frac{p_2}{z_1})^2\cdot v(T-\tau)] \\ &= (p_c + \frac{p_r}{T})[(1 - q(\frac{p_1\gamma}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)}))^2 \\ &+ q(\frac{p_1\gamma}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2(v-1) + q(\frac{p_1}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2 v(\tau-1) \\ &+ q(\frac{p_2}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2 v(T-\tau)] \end{aligned}$$

Plugging in the asymptotics from Assumption 2.1.3, the above becomes

$$p_c[(1 - q(\frac{p_1\gamma}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)}))^2 + q(\frac{p_1\gamma}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2(v-1)] \pm O(\frac{1}{T}) \tag{2.23}$$

Case 2: $\texttt{topic}(\tilde{w}_j) \in \{t_1, \cdots, t_\tau\}\backslash\{\texttt{topic}(w_j)\}$ When $\tilde{w}_j$ after masking belongs to a different topic from that of the correct token w_j, but still a topic existing in $\boldsymbol{w}$. (This happens with probability $\frac{p_r(\tau-1)}{T}$)

$\left[\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}})\right]_{:j}$ is the same as (2.22).

Hence the loss is

$$\begin{aligned} &p_r\frac{\tau-1}{T}[(1 - q(\frac{p_1}{z_1}))^2 + q(\frac{p_1\gamma}{z_1})^2\cdot v + q(\frac{p_1}{z_1})^2\cdot(v(\tau-1)-1) + q(\frac{p_2}{z_1})^2\cdot v(T-\tau)] \\ &= p_r\frac{\tau-1}{T}[(1 - q(\frac{p_1}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2 + q(\frac{p_1\gamma}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2\cdot v \\ &+ q(\frac{p_1}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2(v(\tau-1)-1) \\ &+ q(\frac{p_2}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2 v(T-\tau)] \end{aligned}$$

Plugging in the asymptotics from Assumption 2.1.3, the above terms vanish.

Case 3: $\texttt{topic}(\tilde{w}_j) \in [T]\backslash\{t_1, \cdots, t_\tau\}$ When $\tilde{w}_j$ after masking belongs to a topic that does not exist in $\boldsymbol{w}$. (This happens with probability $p_r(1 - \frac{\tau}{T})$)

By (2.20),

$$\left[\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}})\right]_{ij} = \begin{cases} \frac{p_2\beta}{z_2}, & \text{if } i = \tilde{w}_j \\ \frac{p_2\alpha}{z_2}, & \text{if } i \neq \tilde{w}_j, \texttt{topic}(i) = \texttt{topic}(\tilde{w}_j) \\ \frac{p_1}{z_2}, & \text{if } \texttt{topic}(i) \neq \texttt{topic}(\tilde{w}_j), \texttt{topic}(i) \in \{t_1, \cdots, t_\tau\} \\ \frac{p_2}{z_2}, & \text{if } \texttt{topic}(i) \neq \texttt{topic}(\tilde{w}_j), \texttt{topic}(i) \notin \{t_1, \cdots, t_\tau\} \end{cases} \tag{2.24}$$

$$\frac{1}{v}\sum_{l\in\text{topic}(i)} \left[\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}})\right]_{lj} = \begin{cases} \frac{1}{v}\left(\frac{p_2\beta}{z_2} + \sum_{l\in\text{topic}(i), l\neq\tilde{w}_j} \frac{p_2\alpha}{z_2}\right) = \frac{1}{v}\left(\frac{p_2\beta+(v-1)p_2\alpha}{z_2}\right) = \frac{p_2\gamma}{z_2}, & \text{if } \texttt{topic}(i) = \texttt{topic}(\tilde{w}_j) \\ \frac{1}{v}\left(\sum_{l\in\text{topic}(i)} \frac{p_1}{z_2}\right) = \frac{p_1}{z_2}, & \text{if } \texttt{topic}(i) \neq \texttt{topic}(\tilde{w}_j), \texttt{topic}(i) \in \{t_1, \cdots, t_\tau\} \\ \frac{1}{v}\left(\sum_{l\in\text{topic}(i)} \frac{p_2}{z_2}\right) = \frac{p_2}{z_2}, & \text{if } \texttt{topic}(i) \neq \texttt{topic}(\tilde{w}_j), \texttt{topic}(i) \notin \{t_1, \cdots, t_\tau\} \end{cases} \tag{2.25}$$

Hence the loss is

$$\begin{aligned} &p_r(1-\frac{\tau}{T})[(1-q(\frac{p_1}{z_2}))^2 + q(\frac{p_1}{z_2})^2\cdot(v\tau-1) + q(\frac{p_2\gamma}{z_2})^2\cdot v + q(\frac{p_2}{z_2})^2\cdot v(T-\tau-1)] \\ &= p_r(1-\frac{\tau}{T})[(1-q(\frac{p_1}{p_2\gamma v + p_1 v\tau + p_2 v(T-\tau-1)}))^2 + q(\frac{p_1}{p_2\gamma v + p_1 v\tau + p_2 v(T-\tau-1)})^2(v\tau-1) \\ &+ q(\frac{p_2\gamma}{p_2\gamma v + p_1 v\tau + p_2 v(T-\tau-1)})^2 v + q(\frac{p_2}{p_2\gamma v + p_1 v\tau + p_2 v(T-\tau-1)})^2 v(T-\tau-1)] \end{aligned}$$

Plugging in the asymptotics from Assumption 2.1.3, the above becomes

$$p_r(1+q(\frac{p_2\gamma}{p_2\gamma v + p_1 v\tau + p_2 v(T-\tau-1)})^2) \tag{2.26}$$

Combining the above cases Adding (2.23) and (2.26), we can see in the asymptotic regime of interest, we have:

$$\begin{aligned} L(\gamma) &= p_c[(1-q(\frac{p_1\gamma}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)}))^2 + q(\frac{p_1\gamma}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2(v-1)] \\ &\quad + p_r(1+q(\frac{p_2\gamma}{p_1\gamma v + p_1 v\tau + p_2 v(T-\tau-1)})^2) \pm O(\frac{1}{T}) \\ &= p_c[(1-\frac{c_4 p_1\gamma}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2 + (\frac{c_4 p_1\gamma}{p_1\gamma v + p_1 v(\tau-1) + p_2 v(T-\tau)})^2(v-1)] \\ &\quad + p_r + p_r(\frac{c_4 p_2\gamma}{p_2\gamma v + p_1 v\tau + p_2 v(T-\tau-1)})^2 \pm O(\frac{1}{T}) \end{aligned}$$

in which the constant c_4 is defined as

- $c_4 \coloneqq \frac{1}{1-(1-p_c)p_m} \in (1,2)$

Plugging in the definition of p_1, p_2 in (2.18), (2.19)

$$\begin{aligned} L(\gamma) &= p_c[(1-\frac{\frac{1}{v\tau}\gamma}{\frac{1}{c_4\tau}\gamma + \frac{1}{c_4\tau}(\tau-1) + \frac{p_m p_r}{T}(T-\tau)})^2 + (\frac{\frac{1}{v\tau}\gamma}{\frac{1}{c_4\tau}\gamma + \frac{1}{c_4\tau}(\tau-1) + \frac{p_m p_r}{T}(T-\tau)})^2(v-1)] \\ &\quad + p_r + p_r(\frac{c_4\frac{p_m p_r}{vT}\gamma}{\frac{p_m p_r}{vT}\gamma v + \frac{1}{c_4} + \frac{p_m p_r}{T}(T-\tau-1)})^2 \pm O(\frac{1}{T}) \\ &= p_c\left[(1-\frac{c_4\gamma}{v\gamma + v(\tau-1) + c_4 p_m p_r v\tau})^2 + (\frac{c_4\gamma}{v\gamma + v(\tau-1) + c_4 p_m p_r v\tau})^2(v-1)\right] \\ &\quad + p_r + p_r(\frac{c_4 p_m p_r\gamma}{p_m p_r\gamma v + (\frac{1}{c_4} + p_m p_r)vT})^2 \pm O(\frac{1}{T}) \end{aligned} \tag{2.27}$$

We will again consider several possible cases for γ in (2.27).

Case 1: When $\gamma \leq \frac{(1+c_4 p_m p_r)(2-c_4)}{2c_4}(\tau - 1)$.

Let c_5 denote the constant:

$$c_5 \coloneqq \frac{(1+c_4 p_m p_r)(2-c_4)}{2c_4}$$

then focusing on this term in the loss (2.27):

$$\begin{aligned}&\frac{c_4\gamma}{v\gamma + v(\tau-1) + c_4 p_m p_r v\tau}\\ &< \frac{c_4\gamma}{v\gamma + v\frac{\gamma}{c_5} + c_4 p_m p_r v\frac{\gamma}{c_5}}\\ &= \frac{c_4}{v + v\frac{1}{c_5} + c_4 p_m p_r v\frac{1}{c_5}}\\ &= \frac{c_4}{v(1+\frac{1+c_4 p_m p_r}{c_5})}\end{aligned}$$

and so

$$L(\gamma) > p_c\left(1 - \frac{c_4}{v(1+\frac{1+c_4 p_m p_r}{c_5})}\right)^2 + p_r \pm o(1) \tag{2.28}$$

Case 2: When $\gamma \geq 100\frac{\frac{1}{c_4}+p_m p_r}{p_m p_r}T$.

then since $\tau = o(T)$ by Assumption 2.1.3:

$$\begin{aligned}\frac{c_4\gamma}{v\gamma + v(\tau-1) + c_4 p_m p_r v\tau} &= \frac{c_4\gamma}{v\gamma} + o(1) = \frac{c_4}{v} + o(1)\\ \frac{c_4\gamma}{v\gamma + v(\tau-1) + c_4 p_m p_r v\tau} &= \frac{c_4\gamma}{v\gamma} + o(1) = \frac{c_4}{v} + o(1)\\ \frac{c_4 p_m p_r\gamma}{p_m p_r\gamma v + (\frac{1}{c_4} + p_m p_r)vT} &\geq \frac{c_4 p_m p_r \gamma}{p_m p_r \gamma v + \frac{1}{100}p_m p_r v\gamma} = \frac{100c_4}{101v}\end{aligned}$$

and therefore plugging into (2.27):

$$\begin{aligned}L(\gamma) &\geq p_c[(1-\frac{c_4}{v} \pm o(1))^2 + (\frac{c_4}{v} \pm o(1))^2(v-1)] + p_r + p_r(\frac{100c_4}{101v})^2 \pm o(1)\\ &= p_c[1 - \frac{2c_4}{v} + \frac{c_4^2}{v^2} + \frac{c_4^2}{v^2}(v-1)] + p_r + p_r(\frac{100c_4}{101v})^2 \pm o(1)\\ &= p_c[1 - \frac{2c_4}{v} + \frac{c_4^2}{v}] + p_r + p_r(\frac{100c_4}{101v})^2 \pm o(1)\\ &= p_c[1 - \frac{c_4(2-c_4)}{v}] + p_r + p_r(\frac{100c_4}{101v})^2 \pm o(1)\end{aligned} \tag{2.29}$$

Case 3: When $\frac{(1+c_4 p_m p_r)(2-c_4)}{2c_4}(\tau-1) < \gamma < 100\frac{\frac{1}{c_4}+p_m p_r}{p_m p_r}T$.

Note that this case is the complement of Case 1 and Case 2 above, and so we have considered all possibilities. We will show that there exists γ in this case such that $L(\gamma)$ is smaller (by an $\Omega(1)$ constant difference) than the lower bound of $L(\gamma)$ proven in Case 1 and Case 2 above, based on which we know $\arg\min L(\gamma)$ cannot lie in Case 1 or Case 2, and thus conclude that $\arg\min L(\gamma)$ is within this case. Specifically, let:

$$\gamma = \sqrt{\tau T}$$

then similar to Case 2, since $\tau = o(T)$ by Assumption 2.1.3:

$$\begin{aligned} \frac{c_4\gamma}{v\gamma + v(\tau-1) + c_4 p_m p_r v\tau} &= \frac{c_4\gamma}{v\gamma} + o(1) = \frac{c_4}{v} + o(1) \\ \frac{c_4\gamma}{v\gamma + v(\tau-1) + c_4 p_m p_r v\tau} &= \frac{c_4\gamma}{v\gamma} + o(1) = \frac{c_4}{v} + o(1) \\ \frac{c_4 p_m p_r \gamma}{p_m p_r \gamma v + (\frac{1}{c_4} + p_m p_r) vT} &= o(1) \end{aligned}$$

and therefore plugging into (2.27):

$$\begin{aligned} L(\gamma) &= p_c[(1 - \frac{c_4\gamma}{v\gamma} \pm o(1))^2 + (\frac{c_4\gamma}{v\gamma} \pm o(1))^2(v-1)] + p_r \pm o(1) \\ &= p_c[(1 - \frac{c_4}{v} \pm o(1))^2 + (\frac{c_4}{v} \pm o(1))^2(v-1)] + p_r \pm o(1) \\ &= p_c[1 - \frac{2c_4}{v} + \frac{c_4^2}{v^2} + \frac{c_4^2}{v^2}(v-1)] + p_r \pm o(1) \\ &= p_c[1 - \frac{2c_4}{v} + \frac{c_4^2}{v}] + p_r \pm o(1) \\ &= p_c[1 - \frac{c_4(2-c_4)}{v}] + p_r \pm o(1) \end{aligned} \tag{2.30}$$

Comparing the above cases
Note that $L(\gamma)$ in Case 3 is strictly smaller than $L(\gamma)$ in Case 1 and Case 2, because:

- Comparing (2.28) and (2.30): $(1 - \frac{c_4}{v(1+\frac{1+c_4 p_m p_r}{c_5})})^2 > 1 - \frac{c_4(2-c_4)}{v}$ because $c_5 \in (0, \frac{(1+c_4 p_m p_r)(2-c_4)}{c_4})$
- Comparing (2.29) and (2.30): in the former, the term $p_r(\frac{100c_4}{101v})^2 > 0$ is the extra constant (of scale $\Omega(1)$, i.e. non-vanishing even under our asymptotic assumptions Assumption 2.1.3) compared with the latter.

Therefore we conclude that

$$\arg\min L(\gamma) \subseteq (\frac{(1+c_4 p_m p_r)(2-c_4)}{2c_4}(\tau-1), 100\frac{\frac{1}{c_4} + p_m p_r}{p_m p_r}T)$$

□

Remark 2.1.9. *In Theorem 2.1.3, we specify some necessary conditions that the optimal γ must satisfy. It is challenging to precisely characterize the optima (to within $o(1)$ error), because doing so may require explicitly writing those smaller scale terms hidden (in $\pm o(1)$) by our asymptotic setting (Assumption 2.1.3). Those smaller scale terms, however, do not affect our analysis, because these $\pm o(1)$ terms cannot reverse the $\Omega(1)$ constant separation between the loss in the above different cases.*

2.1.7.8 Optimal attention weights (when freezing diagonal $\boldsymbol{W}^V$)

Our Stage-2 analysis on the optimal attention weights ((2.5)) is based on freezing $\boldsymbol{W}^V$ to be the Stage-1 optima characterized in Theorem 2.1.2. Notably, in Theorem 2.1.2, the uniqueness of the optima (i.e. a clean block-wise pattern) crucially depends on the L_2 regularization. Indeed, as we prove in Theorem 2.1.4, without the regularization, there is a family of optima (depending on a series of free constants) all of which can encode the topic structure.

Among these alternative optima, we are particularly interested in a special case — one that has a *diagonal* pattern. This type of diagonally structured $\boldsymbol{W}^V$ often occurs when we train the single-layered transformer model without L_2 regularization.

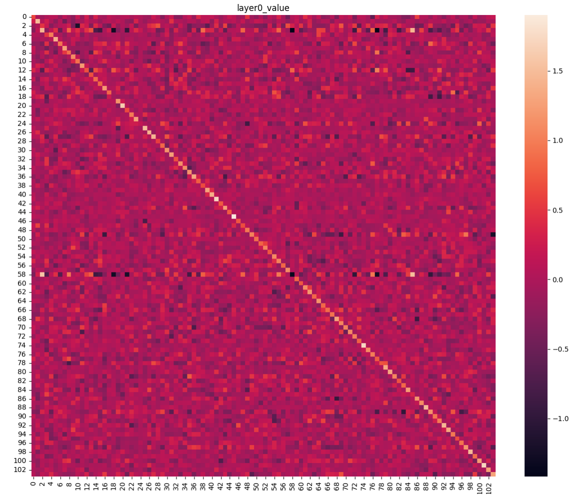


Figure 2.10: Without L_2 regularization, the trained $\boldsymbol{W}^V$ sometimes shows a *diagonal* pattern, which is a special case of the family of optima characterized in Theorem 2.1.4.

Motivated by this empirical observation, we formally define the particular optima from Theorem 2.1.4 that is a diagonal pattern.

Definition 2.1.2 (diagonal $\boldsymbol{W}^V$)**.** *The diagonal optima of $\boldsymbol{W}^V$, denoted as $\boldsymbol{D}^V$, is the only matrix in $\mathbb{R}^{(Tv+1)\times(Tv+1)}$ that satisfies both $\boldsymbol{D}^V \in \arg\min L(\boldsymbol{W}^V)$ (in Theorem 2.1.4) and*

$$\forall i, j \in \{1, \cdots, Tv\},\ \boldsymbol{W}^V_{ij} = 0 \text{ if } i \neq j$$

Corresponding to this case, we provide an analysis on the Stage-2 optimal attention weights, which shows a very interesting different behavior from the result in Theorem 2.1.3 (for block-wise $\boldsymbol{W}^V$).

Theorem 2.1.5 (optimal attention weights when freezing diagonal $\boldsymbol{W}^V$)**.** *Suppose the data distribution follows the topic modeling assumption in Section 2.1.1 and Assumption 2.1.1. Suppose we train a single layer transformer given by (2.7) with $\boldsymbol{b}^{pred} = 0$ and $\boldsymbol{W}^V$ frozen to $\boldsymbol{D}^V$ in Definition 2.1.2, under masked language modeling objective ((2.1)) with the squared loss ((2.3)), under Assumption 2.1.2, Assumption 2.1.3, and Assumption 2.1.4. Then, the optimal (α, β) satisfy:*

$$\lambda_3 \tau < \beta < \lambda_4 T$$
$$\alpha < \lambda_5 \beta$$

in which the constants

$$\lambda_3 \coloneqq \frac{1-(1-p_c)p_m + p_m p_r}{100} \cdot v$$
$$\lambda_4 \coloneqq \frac{1-(1-p_c)p_m}{\sqrt{v-1} - 2 + (1-p_c)p_m} \cdot \frac{1-(1-p_c-p_r)p_m}{p_m p_r} \cdot v$$
$$\lambda_5 \coloneqq \frac{1}{(v-1)(1-(1-p_c)p_m)}$$

Proof. Following the same steps leading to (2.21), define:

$$L(\alpha,\beta) \coloneqq \frac{1}{p_m N} \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\boldsymbol{X}}} \mathbb{E}_M \left[\sum_{j \in M, \tilde{w}_j \neq \texttt{[MASK]}} \| \boldsymbol{D}^V \tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}})_{:j} - \boldsymbol{X}_{:j} \|_2^2 \right] \tag{2.31}$$

Note that $\forall \boldsymbol{y} \in \mathbb{R}^{Tv+1}$

$$(\boldsymbol{D}^V \boldsymbol{y})_i = \begin{cases} 0, & i = 0 \\ q(y_i), & i \in \{1, \cdots, Tv+1\} \end{cases}$$

in which $q(x) \coloneqq \frac{1}{1-(1-p_c)p_m} x - \frac{p_m p_r}{(1-(1-p_c)p_m)Tv}$

In our context, we will consider $\boldsymbol{y} = \tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}})_{:j}$ in $L(\alpha, \beta)$ above.

For a document $\boldsymbol{w}$ which contains topics $t_1, \cdots, t_\tau \in [T]$, there are the following cases:

Case 1: $\tilde{w}_j = w_j$ When $\tilde{w}_j$ after masking is the correct token w_j. (This happens with probability $p_c + \frac{p_r}{vT}$)

By (2.20),

$$\left[\tilde{\boldsymbol{X}} A(\tilde{\boldsymbol{X}})\right]_{ij} = \begin{cases} \frac{p_1 \beta}{z_1}, & \text{if } i = \tilde{w}_j \\ \frac{p_1 \alpha}{z_1}, & \text{if } i \neq \tilde{w}_j, \texttt{topic}(i) = \texttt{topic}(\tilde{w}_j) \\ \frac{p_1}{z_1}, & \text{if } \texttt{topic}(i) \neq \texttt{topic}(\tilde{w}_j), \texttt{topic}(i) \in \{t_1, \cdots, t_\tau\} \\ \frac{p_2}{z_1}, & \text{if } \texttt{topic}(i) \neq \texttt{topic}(\tilde{w}_j), \texttt{topic}(i) \notin \{t_1, \cdots, t_\tau\} \end{cases} \tag{2.32}$$

Recall that the label is $\boldsymbol{X}_{:j} = \begin{cases} 1, & i = w_j \\ 0, & i \in \{0, \cdots, Tv\} \backslash w_j \end{cases}$

Hence the contribution to the loss from token $\tilde{w}_j$ is

$$\begin{aligned} &(p_c + \frac{p_r}{vT})[\left(1 - q\left(\frac{p_1\beta}{z_1}\right)\right)^2 + q(\frac{p_1\alpha}{z_1})^2 \cdot (v-1) + q(\frac{p_1}{z_1})^2 \cdot v(\tau-1) + q(\frac{p_2}{z_1})^2 \cdot v(T-\tau)] \\ &= (p_c + \frac{p_r}{vT})[(1 - q(\frac{p_1\beta}{\beta p_1 + \alpha p_1(v-1) + p_1 v(\tau-1) + p_2 v(T-\tau)}))^2 \\ &+ q(\frac{p_1\alpha}{\beta p_1 + \alpha p_1(v-1) + p_1 v(\tau-1) + p_2 v(T-\tau)})^2 (v-1) \\ &+ q(\frac{p_1}{\beta p_1 + \alpha p_1(v-1) + p_1 v(\tau-1) + p_2 v(T-\tau)})^2 v(\tau-1) \\ &+ q(\frac{p_2}{\beta p_1 + \alpha p_1(v-1) + p_1 v(\tau-1) + p_2 v(T-\tau)})^2 v(T-\tau)] \end{aligned}$$

Plugging in the asymptotics from Assumption 2.1.3, the above becomes

$$\begin{aligned} &p_c[(1-q(\frac{p_1\beta}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)}))^2 \\ &\quad +q(\frac{p_1\alpha}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)})^2(v-1)\pm O(\frac{1}{\tau})] \end{aligned} \tag{2.33}$$

Case 2: $\tilde{w}_j \neq w_j, \texttt{topic}(\tilde{w}_j)=\texttt{topic}(w_j)$ When $\tilde{w}_j$ after masking is not the correct token but belongs to the same topic as the correct token w_j. (This happens with probability $\frac{p_r}{T}(1-\frac{1}{v})$)

$\left[\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}})\right]_{:j}$ is the same as (2.32).

Hence the loss is

$$\begin{aligned} &\frac{p_r}{T}(1-\frac{1}{v})[(1-q(\frac{p_1\alpha}{z_1}))^2+q(\frac{p_1\beta}{z_1}))^2+q(\frac{p_1\alpha}{z_1})^2\cdot(v-2)+q(\frac{p_1}{z_1})^2\cdot v(\tau-1)+q(\frac{p_2}{z_1})^2\cdot v(T-\tau)] \\ =&\frac{p_r}{T}(1-\frac{1}{v})[(1-q(\frac{p_1\alpha}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)}))^2 \\ &+q(\frac{p_1\beta}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)})^2 \\ &+q(\frac{p_1\alpha}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)})^2(v-2) \\ &+q(\frac{p_1}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)})^2v(\tau-1) \\ &+q(\frac{p_2}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)})^2v(T-\tau)] \end{aligned}$$

Plugging in the asymptotics from Assumption 2.1.3, the above terms vanish.

Case 3: $\texttt{topic}(\tilde{w}_j)\in\{t_1,\cdots,t_\tau\}\backslash\{\texttt{topic}(w_j)\}$ When $\tilde{w}_j$ after masking belongs to a different topic from that of the correct token w_j, but still a topic existing in $\boldsymbol{w}$. (This happens with probability $\frac{p_r(\tau-1)}{T}$)

$\left[\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}})\right]_{:j}$ is the same as (2.32).

Hence the loss is

$$\begin{aligned} &p_r\frac{\tau-1}{T}[(1-q(\frac{p_1}{z_1}))^2+q(\frac{p_1\beta}{z_1}))^2+q(\frac{p_1\alpha}{z_1})^2\cdot(v-1)+q(\frac{p_1}{z_1})^2\cdot(v(\tau-1)-1)+q(\frac{p_2}{z_1})^2\cdot v(T-\tau)] \\ =&p_r\frac{\tau-1}{T}[(1-q(\frac{p_1}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)})^2 \\ &+q(\frac{p_1\beta}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)})^2 \\ &+q(\frac{p_1\alpha}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)})^2(v-1) \\ &+q(\frac{p_1}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)})^2(v(\tau-1)-1) \\ &+q(\frac{p_2}{\beta p_1+\alpha p_1(v-1)+p_1v(\tau-1)+p_2v(T-\tau)})^2v(T-\tau)] \end{aligned}$$

Plugging in the asymptotics from Assumption 2.1.3, the above terms vanish.

Case 4: $\texttt{topic}(\tilde{w}_j)\in[T]\backslash\{t_1,\cdots,t_\tau\}$ When $\tilde{w}_j$ after masking belongs to a topic that does not exist in $\boldsymbol{w}$. (This happens with probability $p_r(1-\frac{\tau}{T})$)

By (2.20),

$$\left[\tilde{\boldsymbol{X}}A(\tilde{\boldsymbol{X}})\right]_{ij} = \begin{cases} \frac{p_2\beta}{z_2}, & \text{if } i = \tilde{w}_j \\ \frac{p_2\alpha}{z_2}, & \text{if } i \neq \tilde{w}_j, \texttt{topic}(i) = \texttt{topic}(\tilde{w}_j) \\ \frac{p_1}{z_2}, & \text{if } \texttt{topic}(i) \neq \texttt{topic}(\tilde{w}_j), \texttt{topic}(i) \in \{t_1, \cdots, t_\tau\} \\ \frac{p_2}{z_2}, & \text{if } \texttt{topic}(i) \neq \texttt{topic}(\tilde{w}_j), \texttt{topic}(i) \notin \{t_1, \cdots, t_\tau\} \end{cases} \tag{2.34}$$

Hence the loss is

$$\begin{aligned} & p_r(1-\frac{\tau}{T})[(1-q(\frac{p_1}{z_2}))^2 + q(\frac{p_1}{z_2})^2 \cdot (v\tau - 1) + q(\frac{p_2\beta}{z_2})^2 + q(\frac{p_2\alpha}{z_2})^2 \cdot (v-1) + q(\frac{p_2}{z_2})^2 \cdot v(T-\tau-1)] \\ = & \; p_r(1-\frac{\tau}{T})[(1-q(\frac{p_1}{\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1)}))^2 \\ + & \; q(\frac{p_1}{\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1)})^2(v\tau - 1) \\ + & \; q(\frac{p_2\beta}{\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1)})^2 + q(\frac{p_2\alpha}{\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1)})^2(v-1) \\ + & \; q(\frac{p_2}{\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1)})^2 v(T-\tau-1)] \end{aligned}$$

Plugging in the asymptotics from Assumption 2.1.3, the above becomes

$$\begin{aligned} p_r[(1 + q(\frac{p_2\beta}{\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1)})^2) \\ + q(\frac{p_2\alpha}{\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1)})^2(v-1)] \pm o(1) \end{aligned} \tag{2.35}$$

Combining the above cases Adding (2.33) and (2.35), we can see in the asymptotic regime of interest:

$$\begin{aligned} L(\alpha,\beta) = & \; p_c[(1-q(\frac{p_1\beta}{\beta p_1 + \alpha p_1(v-1) + p_1 v(\tau-1) + p_2 v(T-\tau)}))^2 \\ & + q(\frac{p_1\alpha}{\beta p_1 + \alpha p_1(v-1) + p_1 v(\tau-1) + p_2 v(T-\tau)})^2(v-1)] \\ & + p_r[(1 + q(\frac{p_2\beta}{\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1)})^2) \\ & + q(\frac{p_2\alpha}{\beta p_2 + \alpha p_2(v-1) + p_1 v\tau + p_2 v(T-\tau-1)})^2(v-1)] \pm o(1) \\ = & \; p_c[(1 - \frac{\frac{1}{v\tau}\beta}{\beta\frac{1}{c_4 v\tau} + \alpha\frac{v-1}{c_4 v\tau} + \frac{1}{c_4} + p_m p_r})^2 + (\frac{\frac{1}{v\tau}\alpha}{\beta\frac{1}{c_4 v\tau} + \alpha\frac{v-1}{c_4 v\tau} + \frac{1}{c_4} + p_m p_r})^2(v-1)] \\ & + p_r[(1 + (\frac{c_4\frac{p_m p_r}{vT}\beta}{\beta\frac{p_m p_r}{vT} + \alpha\frac{p_m p_r(v-1)}{vT} + \frac{1}{c_4} + p_m p_r})^2) + (\frac{c_4\frac{p_m p_r}{vT}\alpha}{\beta\frac{p_m p_r}{vT} + \alpha\frac{p_m p_r(v-1)}{vT} + \frac{1}{c_4} + p_m p_r})^2(v-1)] \pm o(1) \end{aligned}$$

in which the constant c_4 is defined as

- $c_4 \coloneqq \frac{1}{1-(1-p_c)p_m} \in (1,2)$ by Assumption 2.1.4.

$$\begin{aligned} L(\alpha,\beta) = & \; p_c[(1 - \frac{c_4\beta}{\beta + (v-1)\alpha + (1+c_4 p_m p_r)v\tau})^2 + (\frac{c_4\alpha}{\beta + (v-1)\alpha + (1+c_4 p_m p_r)v\tau})^2(v-1)] \\ & + p_r[(1 + (\frac{c_4\beta}{\beta + (v-1)\alpha + (\frac{1}{c_4 p_m p_r}+1)vT})^2) + (\frac{c_4\alpha}{\beta + (v-1)\alpha + (\frac{1}{c_4 p_m p_r}+1)vT})^2(v-1)] \pm o(1) \end{aligned} \tag{2.36}$$

We will again consider several possible cases for α, β in (2.36).

- **Case 1,** $\beta \leq \frac{\mathbf{1+c_4p_mp_r}}{\mathbf{100c_4}}\mathbf{v}\tau$: then $\frac{c_4\beta}{\beta+(v-1)\alpha+(1+c_4p_mp_r)v\tau} \leq \frac{c_4\beta}{\beta+100c_4\beta} < \frac{1}{100}$, and hence

$$L(\alpha,\beta) \geq p_c(1-\frac{1}{100})^2 + p_r \pm o(1)$$

- **Case 2,** $\beta > \frac{\mathbf{1+c_4p_mp_r}}{\mathbf{100c_4}}\mathbf{v}\tau$: we have the following subcases:
  - If $\alpha \geq \frac{c_4}{v-1}\beta$, then $\frac{c_4\beta}{\beta+(v-1)\alpha+(1+c_4p_mp_r)v\tau} < \frac{c_4\beta}{\beta+(v-1)\alpha} \leq \frac{c_4\beta}{\beta+c_4\beta} < \frac{c_4}{1+c_4}$, and hence by (2.36) $L(\alpha,\beta) \geq p_c(1-\frac{c_4}{1+c_4})^2 + p_r \pm o(1)$.
  - If $\alpha < \frac{c_4}{v-1}\beta$, then $\frac{c_4\beta}{\beta+(v-1)\alpha+(\frac{1}{c_4p_mp_r}+1)vT} > \frac{c_4\beta}{\beta+c_4\beta+(\frac{1}{c_4p_mp_r}+1)vT}$
    - If $\beta \geq c_7(\frac{1}{c_4p_mp_r}+1)vT$ (for some constant $c_7 := \frac{1}{c_4(\sqrt{v-1}-\frac{1}{c_4}-1)}$), then $\frac{c_4\beta}{\beta+(v-1)\alpha+(\frac{1}{c_4p_mp_r}+1)vT} > \frac{c_4\beta}{\beta+c_4\beta+(\frac{1}{c_4p_mp_r}+1)vT} \geq \frac{c_4\beta}{\beta+c_4\beta+\frac{1}{c_7}\beta} = \frac{c_4}{1+c_4+\frac{1}{c_7}}$, and hence $L(\alpha,\beta) > p_r[1+(\frac{c_4}{1+c_4+\frac{1}{c_7}})^2] \pm o(1)$
    - If $\beta < c_7(\frac{1}{c_4p_mp_r}+1)vT$: note that this case is the complement of all cases (and subcases) above, and so we have considered all possibilities. We will show that there exists (α,β) in this case such that $L(\alpha,\beta)$ is smaller (by an $\Omega(1)$ constant difference) than the lower bound of $L(\alpha,\beta)$ proven in all cases above, based on which we know $\arg\min L(\alpha,\beta)$ cannot lie in any of the above cases, and thus conclude that $\arg\min L(\alpha,\beta)$ is within this case. Specifically: let $\alpha = \sqrt{\tau T}$ and $\beta = \frac{v-1}{c_4-1}\alpha = \frac{v-1}{c_4-1}\sqrt{\tau T}$, then

$$\frac{c_4\beta}{\beta+(v-1)\alpha+(1+c_4p_mp_r)v\tau} = \frac{c_4\frac{v-1}{c_4-1}}{\frac{v-1}{c_4-1}+(v-1)} \pm o(1) = 1 \pm o(1)$$
$$\frac{c_4\alpha}{\beta+(v-1)\alpha+(1+c_4p_mp_r)v\tau} = \frac{c_4}{\frac{v-1}{c_4-1}+(v-1)} \pm o(1) = \frac{c_4-1}{v-1} \pm o(1)$$
$$\frac{c_4\beta}{\beta+(v-1)\alpha+(\frac{1}{c_4p_mp_r}+1)vT} = o(1)$$
$$\frac{c_4\alpha}{\beta+(v-1)\alpha+(\frac{1}{c_4p_mp_r}+1)vT} = o(1)$$

Plugging into (2.36):

$$\begin{aligned} L(\alpha,\beta) &= p_c[(1-(1\pm o(1)))^2 + (\frac{c_4-1}{v-1} \pm o(1))^2(v-1)] + p_r[(1+(o(1))^2)+(o(1))^2(v-1)] \pm o(1) \\ &= p_c[\frac{(c_4-1)^2}{v-1}] + p_r \pm o(1) \\ &< p_c\frac{1}{v-1} + p_r \pm o(1) \end{aligned}$$

  Note that this is smaller than all previous cases, because
    - $\frac{1}{v-1} < (1-\frac{1}{100})^2$ since v is a large finite constant (see Assumption 2.1.3 and Assumption 2.1.4).
    - $\frac{1}{v-1} < (1-\frac{c_4}{1+c_4})^2$ since v is a large finite constant (see Assumption 2.1.3 and Assumption 2.1.4).
    - $\frac{1}{v-1} < (\frac{c_4}{1+c_4+\frac{1}{c_7}})^2$ by the definition of c_7 above.

Therefore, we conclude that all $\alpha,\beta > 0$ that minimize $L(\alpha,\beta)$ must satisfy

$$\begin{aligned} \frac{1+c_4p_mp_r}{100c_4}v\tau <&\beta < c_7(\frac{1}{c_4p_mp_r}+1)vT \\ \alpha <& \frac{c_4}{v-1}\beta \end{aligned}$$

□

Remark 2.1.10. *Remark 2.1.9 applies to this proof too.*

2.1.8 Related Works

One line of prior works explain the success of transformers by empirically showing that the components (e.g. attention heads) of a trained model (e.g. BERT Devlin et al., 2019), contain abundant information for solving a wide range of "probing" tasks, across syntax and semantics (Hewitt & Manning, 2019; Clark et al., 2019; Tenney et al., 2019; Hewitt & Liang, 2019; Kovaleva et al., 2019; Belinkov, 2022), or through other approaches involving the attention weights (Vig & Belinkov, 2019; Htut et al., 2019; Sun & Marasović, 2021). Our result also formalizes some relevant intuitions given in Elhage et al. (2021), such as embedding layer capturing some bigram statistics. In topic modeling distribution, such "bigram statistics" translates to co-occurrence in a document.

Recent works start to combine theoretical constructions and controlled experiments to justify the expressive power of transformers through the lens of Turing completeness (Bhattamishra et al., 2020b), function approximation (Yun et al., 2020), representing formal languages (Bhattamishra et al., 2020a; Ebrahimi et al., 2020; Yao et al., 2021; Liu et al., 2023a), learning abstract algebraic operations (Zhang et al., 2022a), statistical sample complexity (Wei et al., 2021; Edelman et al., 2022), and learning optimal latent representation (Zhang et al., 2023c). Methodologically, we join a long line of works that characterize the capacity of neural network models by assessing their abilities in learning some simple models of the data (Siegelmann & Sontag, 1992; Gers & Schmidhuber, 2001; Weiss et al., 2018; Suzgun et al., 2019; Merrill, 2019; Hewitt et al., 2020; Li & Risteski, 2021; Yao et al., 2021; Zhang et al., 2022a; Liu et al., 2023a). Our work extends this line of works, and in particular, our results indicate that there may be multiple reasonable *representational* optima, which calls for formally analyzing the training dynamics to gain deeper understanding of what the model actually learns from such data distributions.

On the optimization side, Nguyen & Salazar (2019); Xiong et al. (2020); Liu et al. (2020); Zhang et al. (2020); Li & Gong (2021) propose algorithmic improvements (often with theoretical motivations) to help stabilize the training process of transformers. Towards explaining the training process of attention-based neural networks, Sun & Lu (2020) analyzes the trends of two quantities that are relevant to model performance and interpretability in text classification setting.

Also relevant to our work, Snell et al. (2021) consider cross-attention in LSTM Seq2Seq models trained on machine-translation settings[5]. By contrast, we focus on self-attention in transformers, and we consider a data distribution inspired by topic models. Notably, they also propose an intuitive simplifying assumption of a two-stage learning process of the attention heads similar to ours (but without theoretical or empirical validation). Our work uses a similar assumption [6] (Section 2.1.4.1). In our work, we validate our version of the two-stage assumption by providing a particular way to initialize the attention weight matrices, along with theoretical intuitions (Section 2.1.6) and empirical validation on synthetic data (Figure 2.4) as well as real data (Figure 2.9), showing that this two-stage process can be a reasonable approximation to the early steps of the real training dynamics of attention-based models under the settings that we analyze.

Recent works started investigating the theory of Transformers training dynamics. Lu et al. (2021) considers a simple text classification task and proves that the dynamics of training a simple attention-based classification model converges to attending to discriminative words through embedding and the inner product of its key and the query. Recent work by Jelassi et al. (2022) theoretically shows how transformers learn the spatial structure of image-type datasets through gradient-descent-based optimization algorithms. In particular, their attention weights depend on the positional encodings only. Different from their works, our result (motivated by studying the semantics in language) focuses on topic modeling distribution that actually ignores the position information, so the attention weights only depend on the "bag of words" (i.e.

[5] Specifically, they consider a data model related to the IBM machine translation model.

[6] We independently proposed the two-stage training of attention heads, and later discovered (Snell et al., 2021) used a similar assumption. Comparison with (Snell et al., 2021) was added during an update of our paper. Moreover, while Snell et al. (2021) is the earliest paper we are aware of that explicitly assumes a two-stage training process specifically for attention heads, we note that similar approaches (more generally, alternating optimization) commonly appear in the optimization literature in a broad variety of settings.

the contents). In that sense, Jelassi et al. (2022) and our work complement each other, since real-world data distribution usually involves a combination of position-dependent and position-independent factors. An interesting future work would be studying how these factors interact during the training process.

Regarding the type of data distribution that we consider, we join a series of works that theoretically reason about the ability of learning under topic-modeling-based distributions (Sontag & Roy, 2011; Awasthi & Risteski, 2015; Arora et al., 2016; Tosh et al., 2021; Luo et al., 2022). In particular, Luo et al. (2022) shows that if a model can achieve low loss on contrastive or mask-prediction objectives, then it can recover topic posterior. However, these prior works do not theoretically analyze the optimization process of the transformer architecture. In fact, model architecture can indeed critically influence the resulting model obtained by masked-prediction-type tasks (see Liu et al. (2022a) who highlight the subtlety of the interaction between the particular form of the task and the model specification). Hence, our analysis extends beyond the scope of these prior works by incorporating the theoretical analysis on the optimization process of transformers trained on topic modeling data distribution. Empirically, Sia et al. (2020); Thompson & Mimno (2020); Meng et al. (2022); Zhang et al. (2022b); Talebpour et al. (2023) analyze topic discovery via clustering the *contextualized* representations produced by pretrained language models. Different from these works, our theory and experiments on token embeddings focus on the convergence of embedding layer *parameters*.

2.1.9 Conclusion

We initiated the study of understanding training dynamics of transformers in the presence of semantic structure captured by a topic model. Interesting directions of future work includes extending the analysis to data distributions that captures "syntactic" structure, e.g. through simple sandboxes like PCFGs. When both the model and the data distributions are complex, it remains a daunting challenge to "disentangle" how the many different aspects of the data (e.g. semantic and syntactic elements) are learned through the different parts of model architecture (e.g. attention, positional encodings, and embeddings).

2.2 How Transformers learn context-free grammar: qualitatively different optima and pitfalls of myopic interpretability heuristics

Section 2.1 theoretically and empirically characterizes the features learned by a simple Transformer when the data distribution is a topic model. In particular, the learned features, such as the embedding, attention value matrix, and the attention scores, intuitively correspond to the topics in the data distribution. Can we expect such straightforward mapping between learned features and key structures in the data to generally hold for other data distributions?

In this section (based on Wen et al. (2023)), we study a different data distribution, namely a simple context-free grammars known as the Dyck grammar (Section 2.2.1). It turned out that, under this distribution, the learned features in Transformers encode the key structures in the data in a much less intuitive way. In particular, even for small (2-layer) Transformers, the solution space is very rich and does not "uniquely and interpretably" encode the grammatical structure in Dyck.

This result has profound implications on the reliability of some common interpretability methods. Interpretability methods aim to understand the algorithm implemented by a trained model (e.g., a Transofmer) by examining various aspects of the model, such as the weight matrices or the attention patterns. Based on a combination of theoretical results and carefully controlled experiments on synthetic data, we take a critical view of methods that exclusively focus on individual parts of the model, rather than consider the network as a whole. Theoretically, we show that the set of models that (exactly or approximately) solve this task satisfy a structural characterization derived from ideas in formal languages (the pumping lemma). We use this characterization to show that the set of optima is qualitatively rich; in particular, the attention pattern of a single layer can be "nearly randomized", while preserving the functionality of the network. We also show via extensive experiments that these constructions are not merely a theoretical artifact: even after severely constraining the architecture of the model, vastly different solutions can be reached via standard training. Thus, interpretability claims based on inspecting individual heads or weight matrices in the Transformer can be misleading. Our results contribute some theoretical clarity to the debate in the literature (Jain & Wallace, 2019; Serrano & Smith, 2019; Rogers et al., 2020; Grimsley et al., 2020; Brunner et al., 2020; Meister et al., 2021) about the (un)reliability of certain interpretability methods.

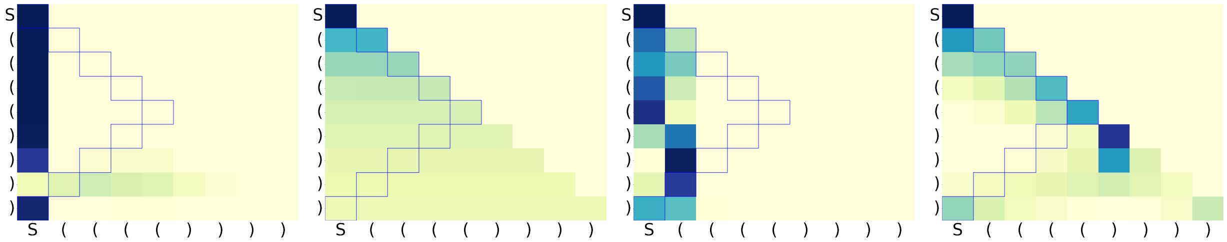
More concretely, we argue that "myopic" interpretability methods, i.e. methods based on examining individual components only, can be provably misleading despite being highly intuitive. We adopt a particular toy setup in which Transformers are trained to generate *Dyck grammars*, a classic type of formal language grammar consisting of balanced parentheses of multiple types. Dyck is a useful sandbox, as it captures properties like long-range dependency and hierarchical tree-like structure that commonly appear in natural and programming language syntax, and has been an object of interest in many theoretical studies of Transofmers (Hahn, 2020; Yao et al., 2021; Liu et al., 2023c;a). Dyck is canonically parsed using a stack-like data structure. Such stack-like patterns (Figure 2.11) have been observed in the attention heads (Ebrahimi et al., 2020), which was later bolstered by mathematical analysis in Yao et al. (2021).

From a representational perspective and via explicit constructions of Transformer weights, recent work (Liu et al., 2023a; Li et al., 2023) show that Transformers are sufficiently expressive to admit very different solutions that perform equally well on the training distribution. Thus, the following questions naturally arise:

(Q1) Do Transformer solutions found empirically match the theoretical constructions given in these representational results (Figure 2.11)? In particular, are interpretable stack-like pattern in Ebrahimi et al. (2020) the norm or the exception in practice?

(Q2) More broadly, can we understand in a principled manner the fundamental obstructions to reliably "reverse engineering" the algorithm implemented by a Transformer by looking at individual attention patterns?

(Q3) Among models that perform (near-)optimally on the training distribution, even if we cannot fully reverse



(a) With Position Embedding

(b) With Position Embedding

(c) Without Position Embedding

(d) Without Position Embedding

Figure 2.11: **Second-layer attention patterns of two-layer Transformers on Dyck**: typical attention patterns do *not* exactly match the intuitively interpretable stack-like pattern prescribed in Ebrahimi et al. (2020); Yao et al. (2021). The blue boxes indicate the locations of the last unmatched open brackets, as they would appear in a stack-like pattern. All models reach $\geq 97\%$ accuracy (defined in Section 2.2.3.2). In the heatmap, darker color indicates larger value.

engineer the algorithm implemented by the learned solutions, can we identify properties that characterize performance beyond the training distribution?

Our contributions. We first prove several theoretical results to provide evidence for why individual components (e.g. attention patterns or weights) of a Transformer should not be expected to be interpretable. In particular, we prove:

- A **perfect balance** condition (Theorem 2.2.1) on the attention pattern that is sufficient and necessary for 2-layer Transformers with a *minimal first layer* (Assumption 2.2.1) to predict optimally on Dyck of *any* length. We then show that this condition permits abundant *non-stack-like* attention patterns that do not necessarily reflect any structure of the task, including *uniform* attentions (Corollary 2.2.1).
- An **approximate balance** condition (Theorem 2.2.2), the *near-optimal* counterpart of the condition above, for predicting on *bounded*-length Dyck. Likewise, non-stack-like attention patterns exist.
- **Indistinguishability from a single component** (Theorem 2.2.3), proved via a *Lottery Ticket Hypothesis* style argument that any Transformer can be approximated by pruning a larger random Transformer, implying that interpretations based exclusively on local components may be unreliable.

We further accompany these theoretical findings with an extensive set of empirical investigations. [7]

Is standard training biased towards interpretable solutions? While both stack-like and non-stack like patterns can process Dyck theoretically, the inductive biases of the architecture or the optimization process may prefer one solution over the other in practice. In Section 2.2.3.2, based on a wide range of Dyck distributions and model architecture ablations, we find that Transformers that generalize near-perfectly in-distribution (and reasonably well out-of-distribution) do *not* typically produce stack-like attention patterns, showing that the results reported in prior work (Ebrahimi et al., 2020) should not be expected from standard training.

Do non-interpretable solutions perform well in practice? Our theory predicts that balanced (or even uniform) attentions suffice for good in- and out-of-distribution generalization. In Section 2.2.3.3, we empirically verify that with standard training, the extent to which attentions are balanced is positively correlated with generalization performance. Moreover, we can guide Transformers to learn more balanced attention by regularizing for the balance condition, leading to better length generalization.

[7] Code is released at https://openreview.net/attachment?id=OitmaxSAUu&name=supplementary_material

2.2.1 Technical setup

Dyck languages A Dyck language (Schützenberger, 1963) is generated by a context-free grammar, where the valid strings consist of balanced brackets of different types (for example, "[()]" is valid but "([)]" is not). Dyck_k denote the Dyck language defined on k types of brackets. The alphabet of Dyck_k is denoted as $[2k] \equiv \{1, 2, \cdots, 2k\}$, where for each type $t \in [k]$, tokens $2t-1$ and $2t$ are a pair of corresponding open and closed brackets. Dyck languages can be recognized by a push-down automaton — by pushing open brackets onto a stack and and popping open brackets when it encounters matching closed brackets. For a string w and $i \le j \in \mathbb{Z}_+$, we use $w_{i:j}$ to denote the substring of w between position i and position j (both ends included). For a valid prefix $w_{1:i}$, the *grammar depth* of $w_{1:i}$ is defined as the depth of the stack after processing $w_{1:i}$:

$$d(w_{1:i}) = \#\text{Open Brackets in } w_{1:i} - \#\text{Closed Brackets in } w_{1:i}.$$

We overload $d(w_{1:i})$ to also denote the grammar depth of the bracket at position i. For example, in each pair of matching brackets, the closing bracket is one depth smaller than the open bracket. We will use $\tau_{i,d}$ to denote a token of type $i \in [2k]$ placed at grammar depth $d \in \mathbb{N}$.

We consider *bounded-depth* Dyck languages following Yao et al. (2021). Specifically, $\mathsf{Dyck}_{k,D}$ is a subset of Dyck_k such that the depth of any prefix of a word is bounded by D,

$$\mathsf{Dyck}_{k,D} := \{w_{1:n} \in \mathsf{Dyck}_k \mid \max_{i\in[n]} d(w_{1:i}) \le D\}. \tag{2.37}$$

While a bounded grammar depth might seem restrictive, it suffices to capture many practical settings. For example, the level of recursion occurring in natural languages is typically bounded by a small constant (Karlsson, 2007; Jin et al., 2018). We further define the *length-N prefix set* of $\mathsf{Dyck}_{k,D}$ as

$$\mathsf{Dyck}_{k,D,N} = \{w_{1:N} \mid \exists n \ge N, w_{N+1:n} \in [2k]^{n-N}, s.t.\ w_{1:n} \in \mathsf{Dyck}_{k,D}\}. \tag{2.38}$$

Our theoretical setup uses the following data distribution $\mathcal{D}_{\mathsf{Dyck}}$:

Definition 2.2.1 (Dyck distribution)**.** *The distribution $\mathcal{D}_{\mathsf{Dyck}}$, specified by $q \in (0,1)$, is defined over $\mathsf{Dyck}_{k,D,N}$ such that $\forall w_{1:N} \in \mathsf{Dyck}_{k,D,N}$,*

$$\begin{aligned}\mathbb{P}(w_{1:N}) \propto & (1/k)^{\#\{i|w_i \text{ is open, } d(w_{1:i})=1\}} \cdot (q/k)^{\#\{i|w_i \text{ is open, } d(w_{1:i})>1\}} \\ & \cdot (1-q)^{\#\{i|w_i \text{ is closed, } d(w_{1:i})<D-1\}}.\end{aligned} \tag{2.39}$$

That is, $q \in (0,1)$ denote the probability of seeing an open bracket at the next position, except for two corner cases: 1) the next bracket has to be open if the current grammar depth is 0 (1 after seeing the open bracket); 2) the next bracket has to be closed if the current grammar depth is D. Note that at any position, there is at most one valid closing bracket.

Training Objectives. Given a model f_θ parameterized by θ, we train with a *next-token prediction* language modeling objective on a given $\mathcal{D}_{\mathsf{Dyck}}$. Precisely, given a loss function $l(\cdot,\cdot) \to \mathbb{R}$, f_θ is trained to minimize the loss function $\min_\theta \mathcal{L}(\theta; \mathcal{D}_{\mathsf{Dyck}})$ with

$$\mathcal{L}(\theta; \mathcal{D}_{\mathsf{Dyck}}) = \mathbb{E}_{w_{1:N}\sim\mathcal{D}_{\mathsf{Dyck}}}\left[\frac{1}{N}\sum_{i=1}^{N} l(f_\theta(w_{1:i-1}), z(w_i))\right] \tag{2.40}$$

in which $z(w_i) \in \{0,1\}^{2k}$ denotes the one-hot embedding of token w_i. We will omit the distribution $\mathcal{D}_{\mathsf{Dyck}}$ when it is clear from the context. We will also consider a ℓ_2-regularized version $\mathcal{L}^{\text{reg}}(\theta) = \mathcal{L}(\theta) + \lambda\frac{\|\theta\|_2^2}{2}$ with parameter $\lambda > 0$.

For our theory, we will consider the mean squared error as the loss function: [8]

$$l \coloneqq l_{sq}(x, z_i) = \|x - z_i\|_2^2. \tag{2.41}$$

In our experiments, we apply the cross entropy loss following common practice.

Transformer Architecture. We consider a general formulation of Transformer in this work: the l-th layer is parameterized by $\theta^{(l)} := \{W_Q^{(l)}, W_K^{(l)}, W_V^{(l)}, \mathrm{param}(\mathrm{g}^{(l)})\} \in \Theta$, where $W_K^{(l)}, W_Q^{(l)} \in \mathbb{R}^{m_a \times m}$, and $W_V^{(l)} \in \mathbb{R}^{m \times m}$ are the key, query, and value matrices of the attention module; $\mathrm{param}(\mathrm{g}^{(l)})$ are parameters of a feed-forward network $\mathrm{g}^{(l)}$, consisting of fully connected layers, (optionally) LayerNorms and residual links. Given $X \in \mathbb{R}^{m \times N}$, the matrix of m-dimensional features on a length-N sequence, the l-th layer of a Transformer computes the function

$$f_l(X; \theta^{(l)}) = \mathrm{g}^{(l)}\Big(\mathrm{LN}\Big(W_V^{(l)} X\, \underbrace{\sigma\Big(\mathcal{C} + (W_K^{(l)} X)^\top (W_Q^{(l)} X)\Big)}_{\text{attention pattern}}\Big) + X\Big), \tag{2.42}$$

where σ is the column-wise softmax operation defined as $\sigma(A)_{i,j} = \frac{\exp(A_{i,j})}{\sum_{k=1}^N \exp(A_{k,j})}$, $\mathcal{C}$ is the causal mask matrix defined as $\mathcal{C}_{i,j} = -\inf \cdot \mathbb{1}[i > j]$ where inf denotes infinity. We call $\sigma\Big(\mathcal{C} + (W_K^{(l)} X)^\top (W_Q^{(l)} X)\Big)$ the *Attention Pattern* of the Transformer layer l. LN represents column-wise LayerNorm operation, whose j_{th} output column is defined as:

$$\mathrm{LN}_C(A)_{:,j} = \frac{\mathcal{P}_\perp A_{:,j}}{\max\{\|\mathcal{P}_\perp A_{:,j}\|_2, C\}}, \mathcal{P}_\perp = \mathcal{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^\top. \tag{2.43}$$

Here $\mathcal{P}_\perp$ denotes the projection orthogonal to the $\mathbf{1}\mathbf{1}^\top$ subspace [9] and C is called the normalizing constant for LayerNorm.

We will further define the *attention output* at the l-th layer as

$$a_l(X; \theta^{(l)}) = W_V^{(l)} X \sigma\Big(\mathcal{C} + (W_K^{(l)} X)^\top (W_Q^{(l)} X)\Big). \tag{2.44}$$

When $C = 0$, we will also consider the *unnormalized attention output* as

$$\tilde{a}_l(X; \theta^{(l)}) = W_V^{(l)} X \tilde{\sigma}\Big(\mathcal{C} + (W_K^{(l)} X)^\top (W_Q^{(l)} X)\Big). \tag{2.45}$$

where $\tilde{\sigma}(A)_{i,j} = \exp(A_{i,j})$ and it holds by definition that $\mathrm{LN}_0(\tilde{a}_l(X; \theta^{(l)})) = \mathrm{LN}_0(a_l(X; \theta^{(l)}))$.

An L-layer Transformer $\mathcal{T}_L$ consists of a composition of L of the above layers, along with a word embedding matrix $W_E \in \mathbb{R}^{m \times 2k}$ and a linear decoding head with weight $W_{\mathrm{Head}} \in \mathbb{R}^{2k \times w}$. When inputting a sequence of tokens into Transformer, we will append a *starting token* $t_\mathcal{S}$ that is distinct from any token in the language at the beginning of the sequence. Let $\mathcal{Z} \in \mathbb{R}^{2k \times (N+1)}$ denote the one-hot embedding of a length-N sequence, then $\mathcal{T}_L$ computes for $\mathcal{Z}$ as

$$\mathcal{T}(\mathcal{Z}) = W_{\mathrm{Head}}\Big[f_L(\cdots (f_1(W_E \mathcal{Z})))\Big]_{1:2k,(N+1)} \tag{2.46}$$

[8] The challenge of applying our theory to cross-entropy loss is that for some prefixes, their grammatical immediate continuations strictly exclude certain tokens in the vocabulary (e.g. "]" cannot immediately follow "{"), so the optimal cross-entropy loss can only be attained if some parameters are set to infinity. However, when label smoothing is added, the optima is finite again, and analysis similar to ours could plausibly apply.

[9] this is just a compact way to write the standard mean subtraction operation

2.2.2 Theory: optimal attention

Many prior works have looked for intuitive interpretations of Transformer solutions by studying the attention patterns of particular heads or some individual components of a Transformer (Clark et al., 2019; Vig & Belinkov, 2019; Dar et al., 2022). However, we show in this section why this methodology can be insufficient even for the simple setting of Dyck. Namely, for Transformers that generalize well on Dyck (both in-distribution and out-of-distribution), neither attention patterns nor individual local components are guaranteed to encode structures specific for parsing Dyck. We further argue that the converse is also insufficient: when a Transformer does produce interpretable attention patterns (suitably formalized), there could be limitations of such interpretation as well, as discussed in Appendix 2.2.4. Together, our results provide theoretical evidence that careful analyses (beyond heuristics) are required when interpreting the components of a learned Transformer.

We focus on Transformers with 2 layers, which are representationally sufficient for processing Dyck (Yao et al., 2021). We will show that even under this simplified setting, attention patterns alone are not sufficient for interpretation. In fact, we will further restrict the set of 2-layer Transformers by requiring the first-layer outputs to only depend on information necessary for processing Dyck:

Assumption 2.2.1 (Minimal First Layer)**.** *We consider 2-layer Transformers with a* minimal first layer f_1*. That is, if $\boldsymbol{Z} \in \mathbb{R}^{2k\times(N+1)}$ denotes the one-hot embeddings of an input sequence $t_{\mathcal{S}}, t_1, \ldots, t_N \in [2k]$, then we assume the $(j+1)_{th}$ column of the output $f_1(W^E\boldsymbol{Z})$ only depends on the type and depth of t_j, $\forall j \in [N]$.*

Assumption 2.2.1 requires the first layer output to depend only on the bracket type and depth, disregarding any other information such as positions; an example of such a layer is given by Yao et al. (2021). The construction of a minimal first layer can vary, hence we *directly parameterize its output* instead:

Definition 2.2.2 (Minimal first layer embeddings)**.** *Given a minimal first layer, $\boldsymbol{e}(\tau_{t,d}) \in \mathbb{R}^m$ denotes its output embedding of $\tau_{t,d}$ for $t \in [2k]$, $d \in [D]$. $\boldsymbol{e}(t_{\mathcal{S}}) \in \mathbb{R}^m$ is the embedding of the starting token.*

It is important to note that while the minimal first layer is a strong condition, it does not weaken our results: We will show that the function class allows for a rich set of solutions, none of which are necessarily interpretable. Relaxing to more complex classes will only expand the solution set, and hence our conclusion will remain valid. See Section 2.2.5.2 for more technical details.

2.2.2.1 Perfect Balance Condition: Ideal Generalization of Unbounded Length

Some prior works have tried to understand the model by inspecting the attention patterns (Ebrahimi et al., 2020; Clark et al., 2019; Vig & Belinkov, 2019). However, we will show that the attention patterns alone are too flexible to be helpful, even for the restricted class of a 2-layer Transformer with a minimal first layer (Assumption 2.2.1) and even on a language as simple as Dyck. In particular, the Transformer only needs to satisfy what we call the *balanced condition*:

Definition 2.2.3 (Balance condition)**.** *A 2-layer Transformer (Equation (2.46)) with a minimal first layer (Assumption 2.2.1 and Definition 2.2.2) is said to satisfy the* balance condition*, if for any $i, j_1, j_2 \in [k]$ and $d', d_1, d_2 \in [D]$,*

$$(\boldsymbol{e}(\tau_{2i-1,d'}) - \boldsymbol{e}(\tau_{2i,d'-1}))^\top (W_K^{(2)})^\top W_Q^{(2)} (\boldsymbol{e}(\tau_{2j_1,d_1}) - \boldsymbol{e}(\tau_{2j_2,d_2})) = 0. \tag{2.47}$$

The following result shows that under minor conditions the balance condition is both necessary and sufficient:

Theorem 2.2.1 (Perfect Balance)**.** *Consider a two-layer Transformer $\mathcal{T}$ (Equation (2.46)) with a minimal first layer (Assumption 2.2.1) and $C = 0$ (Equation (2.43)). Let $\mathcal{O}$ denote the* optimal prediction *scenario, that is, when the first layer embeddings $\{\boldsymbol{e}(\tau_{i,d})\}_{d\in[D],i\in[2k]}$ (Definition 2.2.2) and second layer parameters $\theta^{(2)}$ satisfy*

$$\theta := \{\boldsymbol{e}(\tau_{i,d})\}_{d\in[D],i\in[2k]}, \theta^{(2)}\} = \arg\min_{\tilde{\theta}} \mathcal{L}(\tilde{\theta}; \mathcal{D}_{\mathsf{Dyck}}), \forall N,$$

where the objective $\mathcal{L}$ is defined in Equation (2.40). Then,

- *Equation (2.47) is a necessary condition of $\mathcal{O}$, if $W_V^{(2)}$ satisfies $\mathcal{P}_\perp W_V^{(2)} \boldsymbol{e}(\tau_{t,d}) \neq 0, \forall t \in [k], d \in [D]$.*
- *Equation (2.47) is a sufficient condition of $\mathcal{O}$, for a construction in which the set of $2k+1$ encodings $\{\boldsymbol{e}(\tau_{2i-1,d}), \boldsymbol{e}(\tau_{2i,d})\}_{i\in[k]} \cup \{\boldsymbol{e}(t_\mathcal{S})\}$ are linearly independent for any $d \in [D]$ and the projection function $\mathrm{g}^{(2)}$ is a 6-layer MLP* [10] *with $O(k^2D^2)$ width.*

Remark: Recall from Equation (2.43) that $\mathcal{P}_\perp$ projects to the subspace orthogonal to $\mathbf{1}\mathbf{1}^\top$. The assumption in the "necessary condition" part of the theorem can be intuitively understood as requiring all tokens to have nonzero contributions to the prediction after the LayerNorm.

Recall that $\boldsymbol{e}(\tau_{2i-1,d'}), \boldsymbol{e}(\tau_{2i,d'-1})$ denote the first-layer outputs for a matching pair of brackets. Intuitively, Equation (2.47) says that since matching brackets should not affect future predictions, their embeddings should balance out each other. The balance condition Equation (2.47) is "perfect" in the sense that for the theorem, the model is required to minimize the loss for any length N; we will see an approximate version which relaxes this in Theorem 2.2.2.

Proof of the necessity of the balance condition. The key idea is reminiscent of the pumping lemma for regular languages. For any prefix p ending with a closed bracket $\tau_{2j,d}$ for $d \geq 1$ and containing brackets of all depths in $[D]$, let p_β be the prefix obtained by inserting β pairs of $\{\tau_{2i-1,d'}, \tau_{2i,d'-1}\}$ for arbitrary $i \in [k]$ and $d' \in [D]$. Denote the *projection of the unnormalized attention output* by

$$u(\tau_{t_1,d_1}, \tau_{t_2,d_2}) := \mathcal{P}_\perp \exp\left(\boldsymbol{e}(\tau_{t_1,d_1})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{t_2,d_2})\right) W_V^{(2)} \boldsymbol{e}(\tau_{t_1,d_1}). \tag{2.48}$$

We ignored the normalization in softmax above, since the attention output will be normalized directly by LayerNorm according to Equation (2.42).

By Equation (2.42), for any $X \in \mathbb{R}^{m\times(N+1)}$ we have that

$$\tilde{a}_2(X;\theta^{(2)}) = \sum_{i=1}^{N+1} \mathcal{P}_\perp \exp\left(X_{1:m,i}^\top (W_K^{(2)})^\top W_Q^{(2)} X_{1:m,(N+1)}\right) W_V^{(2)} X_{1:m,(N+1)}.$$

Choosing X as the output of the first layer when the input is p_β, it holds that there exists a vector $v \in \mathbb{R}^m$ such that for any $\beta \in \mathbb{N}$, the next-token logits given by Transformer $\mathcal{T}$ are

$$\mathcal{T}(p_\beta) = W_{\text{Head}} g^{(2)} \left(\frac{v + \beta\left(u(\tau_{2j,d}, \tau_{2i,d'-1}) + u(\tau_{2j,d}, \tau_{2i-1,d'})\right)}{\left\|v + \beta\left(u(\tau_{2j,d}, \tau_{2i,d'-1}) + u(\tau_{2j,d}, \tau_{2i-1,d'})\right)\right\|_2} + \boldsymbol{e}(\tau_{2j,d}) \right). \tag{2.49}$$

The proof proceeds by showing a contradiction. Suppose $u(\tau_{2j,d}, \tau_{2i,d'-1}) + u(\tau_{2j,d}, \tau_{2i-1,d'}) \neq 0$. Based on the continuity of the projection function and the LayerNorm Layer, we can show that $\lim_{\beta\to\infty} \mathcal{T}(p_\beta)$ depend only on the depths d, d' and types $2j, 2i-1, 2i$. However, these are not sufficient to determine the next-token probability from p_β, since the latter depends on the type of the last unmatched open bracket in p. This contradicts the assumption that the model can minimize the loss for any length N. Hence we must have

$$u(\tau_{2j,d}, \tau_{2i,d'-1}) + u(\tau_{2j,d}, \tau_{2i-1,d'}) = 0. \tag{2.50}$$

Finally, as we assumed that $\mathcal{P}_\perp W_V^{(2)} e(\tau_{t,d}) \neq 0$, we conclude that

$$\left(e\left(\tau_{2i-1,d'}\right) - e\left(\tau_{2i,d'-1}\right)\right)^\top (W_K^{(2)})^\top W_Q^{(2)} e\left(\tau_{2j+1,d}\right) = \ln\left(\frac{\|\mathcal{P}_\perp W_V e\left(\tau_{2i,d'-1}\right)\|_2}{\|\mathcal{P}_\perp W_V e\left(\tau_{2i-1,d'}\right)\|_2}\right),$$

where the right hand side is independent of j, d, concluding the proof for necessity. The proof of sufficiency are given in Appendix 2.2.5.1. □

[10] In the construction, we first use 4 layers to convert the input of the projection function to a triplet indicating the type and depth of the last token and the type of the last unmatched bracket when the last token is a closed bracket. We then use another 2 layers to predict the next token probability based on the triplet. This construction is likely improvable.

Note that the perfect balance condition is an orthogonal consideration to interpretability. For example, even the uniform attention satisfies the condition and can solve Dyck: [11]

Corollary 2.2.1. *There exists a 2-layer Transformer with uniform attention and no positional embedding (but with causal mask and a starting token [12]) that generates the Dyck language of arbitrary length.*

Since uniform attention patterns are hardly reflective of any structure of Dyck, Corollary 2.2.1 proves that attention patterns can be oblivious about the underlying task, violating the "faithfulness" criteria for an interpretation (Jain & Wallace, 2019). We will further show in Section 2.2.4.1 that empirically, seemingly structured attention patterns may not accurately represent the natural structure of the task.

2.2.2.2 Approximate Balance Condition For Finite Length Training Data

Theorem 2.2.1 assumes the model reaches the optimal loss for Dyck prefixes of any length. However, in practice, due to finite samples and various sources of randomness, training often does not end exactly at a population optima. In this case, the condition in Theorem 2.2.1 is not precisely met. However, even for models that *approximately* meet those conditions, we will prove that when the second-layer projection function $\mathrm{g}^{(2)}$ is Lipschitz, a similar condition as in Equation (2.50) is still necessary.

We will show this by bounding the amount of deviations from the perfect balance. The idea is that for two long prefixes that differ in only the last open bracket, correct next token prediction requires the Transformer outputs on these prefixes to be sufficiently different, hence the part irrelevant to the prediction (i.e. matched brackets) should not have a large contribution.

To formalize this intuition, we define two quantities: 1) $S_{d,d',i,j}$ which measures the effect from one matching pair, and 2) $P_{d,j}$ which measures the effect on the last position from all tokens in a prefix.

Let u be defined as in Equation (2.48). $S_{d,d',i,j}$ is defined as

$$S_{d,d',i,j}[\theta^{(2)}] = u(\tau_{2j,d}, \tau_{2i,d'-1}) + u(\tau_{2j,d}, \tau_{2i-1,d'}), \tag{2.51}$$

which measures how much a matching pair of brackets $(\tau_{2i,d'-1}, \tau_{2i-1,d'})$ changes the input to the LayerNorm upon seeing the last token $\tau_{2j,d}$. Note that under the perfect balance condition, $S_{d,d',i,j}[\theta^{(2)}] = 0$ by Equation (2.50).

The second quantity $P_{d,j}[\theta^{(2)}]$ is defined via an intermediate quantity $Q(2j, d, \tilde{\boldsymbol{t}})$: for any $i \in [k], d \in [D]$ and a length-$(d-1)$ prefix $\tilde{\boldsymbol{t}} \in [2k]^{d-1}$, $Q(i, d, \tilde{\boldsymbol{t}})$ is defined as

$$\begin{aligned} Q(i, d, \tilde{\boldsymbol{t}}) := u(\tau_{2i,d-1}, t_{\mathcal{S}}) + \sum_{1 \le d' < d} u(\tau_{2i,d-1}, \tau_{\tilde{\boldsymbol{t}}_{d'},d'}) \\ + u(\tau_{2i,d-1}, \tau_{2i-1,d}) + u(\tau_{2i,d-1}, \tau_{2i,d-1}), \end{aligned} \tag{2.52}$$

where $\tilde{\boldsymbol{t}}_{d'}$ denotes the d'_{th} entry of $\tilde{\boldsymbol{t}}$. Intuitively, $Q(i, d, \tilde{\boldsymbol{t}})$ denotes the unnormalized second-layer attention output at the last position, given the input sequence $\tilde{\boldsymbol{t}} \oplus \tau_{2i-1,d}\tau_{2i,d-1}$, [13]

For results in this subsection, it suffices to consider prefixes consisting only of open brackets. Let $\boldsymbol{t} := \arg\min_{\tilde{\boldsymbol{t}} \in \{2i-1\}_{i\in[k]}^{d-1}} \|Q(2j, d, \tilde{\boldsymbol{t}})\|_2$, and let $\boldsymbol{t}'$ denote the prefix that minimizes $\|Q(2j, d, \tilde{\boldsymbol{t}})\|_2$ subject to the constraint that $\boldsymbol{t}'$ differs from $\boldsymbol{t}$ at the last (i.e. $(d-1)_{th}$) position, i.e.

$$\boldsymbol{t}' = \arg \min_{\tilde{\boldsymbol{t}}' \in \{2i-1\}_{i\in[k]}^{d-1}, \boldsymbol{t}'_{d-1} \neq \boldsymbol{t}_{d-1}} Q(2j, d, \tilde{\boldsymbol{t}}').$$

[11] This is verified empirically: the uniform-attention models have attention weights fix to 0 and are to fit the distribution almost perfectly (> 99% accuracy).

[12] Here the starting token is necessary because otherwise, the Transformer with uniform attention will have the same outputs for prefix p and prefix $p \oplus p$, in which $\oplus$ denotes concatenation, i.e. $p \oplus p$ means the same string p repeated twice.

[13] We use $s \oplus t$ to denote the concatenation of two strings s, t, same as in Equation (2.55)-(2.56), and use $\tau_i\tau_j$ to denote the concatenation of two tokens τ_i, τ_j.

Such choices of $\boldsymbol{t}, \boldsymbol{t}'$ guarantees that the two prefixes differ at the last open bracket and hence must have different next-word distributions. Finally, define

$$P_{d,j}[\bar{\theta}^{(2)}] = \|Q(2j, d, \boldsymbol{t}')\|_2. \tag{2.53}$$

In the following theorem, $P_{d,j}$ will be used as a quantity that will denote an upper bound on $S_{d,d',i,j}[\theta^{(2)}]$, meaning that the model should not be sensitive to the insertion of a matching pair of brackets.

Theorem 2.2.2 (Approximate Balance)**.** *Consider a 2-layer Transformer $\mathcal{T}$ (Equation (2.46)) with a minimal first layer (Assumption 2.2.1) and a γ-Lipschitz $\mathrm{g}^{(2)}$ for $\gamma > 0$, trained on sequences of length N with the mean squared loss (Equation (2.41)).*

Suppose the loss is approximately optimal, precisely, the set of second-layer weights $\bar{\theta}_N^{(2)}$ satisfies

$$\mathcal{L}(\mathcal{T}[\bar{\theta}_N^{(2)}], \mathcal{D}_{\mathsf{Dyck}}) \leq (\frac{q(1-q)}{k^2})^N \epsilon$$

for every positive integer $N > 8D$ and sufficiently small $\epsilon > 0$. Then, there exists a constant $C_{\gamma,\epsilon,D}$, such that $\forall 0 \leq d' \leq D, 1 \leq d \leq D, i, j \in [k]$, it holds that

$$\|S_{d,d',i,j}[\bar{\theta}_N^{(2)}]\| \leq \frac{C_{\gamma,\epsilon,D}}{N} P_{d,j}[\bar{\theta}_N^{(2)}]. \tag{2.54}$$

Intuitively, Theorem 2.2.2 states that when the loss $\mathcal{L}(\theta)$ is sufficiently small, $S_{d,d',i,j}[\theta^{(2)}]$ must be small relative to $P_{d,j}[\bar{\theta}_N^{(2)}]$. Inequality 2.54 can be interpreted as a relaxation of Equation (2.50), which is equivalent to $S_{d,d',i,j}[\theta^{(2)}] = 0$. The proof of Theorem 2.2.2 shares a similar intuition as Theorem 2.2.1 and is given in Section 2.2.5.4.

A direct corollary of Theorem 2.2.2 additionally considers weight decay as well, in which case approximate balance condition still holds, as the regularization strength goes to 0:

Corollary 2.2.2. *Consider the setting where a Transformer with a fixed minimal first layer is trained to minimize $\mathcal{L}_\lambda^{reg} = \mathcal{L}_\theta(x) + \lambda \frac{\|\theta\|_2^2}{2}$, which is the squared loss with λ weight decay. Suppose $\mathrm{g}^{(1)}$ of the Transformer is a 2-layer fully connected network and $\mathrm{g}^{(2)}$ of the Transformer is a 6-layer fully connected network. Then, there exists constant $C > 0$, such that if a set of parameters $\theta_{\lambda,N}$ minimizes $\mathcal{L}_\lambda^{reg}$, then it holds $\forall 0 \leq d' \leq D, 1 \leq d \leq D, i, j \in [k]$ that,*

$$\forall N, \exists \lambda_N, \text{ such that } \forall \lambda \in [0, \lambda_N], S_{d,d',i,j}[\theta_{\lambda,N}^{(2)}] \leq \frac{C}{N} P_{d,i}[\theta_{\lambda,N}^{(2)}].$$

The proof relies on the continuity of $\mathcal{L}_\lambda^{\text{reg}}$ w.r.t. λ, whose detail is at the end of Section 2.2.5.4.

2.2.2.3 Theory: indistinguishability from a single weight matrix

Another line of interpretability works involves inspecting the weight matrices of the model (Li et al., 2016; Dar et al., 2022; Eldan & Li, 2023). Some of the investigations are done locally, neglecting the interplay between different parts of the model. Our result in this section shows that from a representational perspective, isolating single weights can also be misleading for interpretability. For this section only, we will assume the linear head W_{Head} is identity for simplicity. To consider the effect of pruning, we will also extend the parameterization of LayerNorm module (Equation (2.43)) as

$$\mathrm{LN}_C[b](A)_{:,j} = b\frac{\mathcal{P}_\perp A_{:,j}}{\max\{\|\mathcal{P}_\perp A_{:,j}\|_2, \epsilon\}} + (1-b)A_{:,j},$$

which corresponds to a weighted residual branch; note that the original LayerNorm corresponds to $\mathrm{LN}_C[1]$. [14] Let $\hat{\theta}$ denote the set of parameters of this extended parameterization.

We define the *nonstructural pruning* [15] as:

[14] This residue link is added for the ease of proof because it is hard to "undo" a LayerNorm. We also note that in standard architecture like GPT-2, there is typically a residual link after LayerNorm similar to here.

[15] This is as opposed to *structural pruning*, which prunes entire rows/columns of weight matrices.

Definition 2.2.4 (Nonstructural pruning)**.** *Under the extended parameterization, a* nonstructural pruning *of a Transformer with parameters $\hat{\theta}$ is a Transformer with the same architecture and parameters $\hat{\theta}'$, so that for any weight matrix W in $\hat{\theta}$, the corresponding matrix W' in $\hat{\theta}'$ satisfies $W'_{i,j} \in \{W_{i,j}, 0\}$, $\forall i, j$.*

To measure the quality of the pruning, define the ϵ-approximation:

Definition 2.2.5 (ϵ-approximation)**.** *Given two metric spaces A, B with the same metric $\|\cdot\|$, a function $f : A \to B$ is an ϵ-approximation of function g with respect to that metric, if and only if,*

$$\forall x \in A, \|f(x) - g(x)\| \leq \epsilon \|x\|.$$

The metric, unless otherwise specified, will be the 2-norm for vectors and the $1,2$-norm for matrices:

Definition 2.2.6. *The $1,2$-norm of a matrix A is the max row norm, i.e. $\|A\|_{1,2} = \max_{i\in[d']} \|A_{:,i}\|_2$.*

With these definitions, we are ready to state the main result of this section:

Theorem 2.2.3 (Indistinguishability From a Single Component)**.** *Consider any L-layer Transformer $\mathcal{T}$ (Equation (2.46)) with embedding dimension m, attention dimension m_a, and projection function $\mathrm{g}^{(l)}$ as 2-layer ReLU MLP with width w, for $l \in [L]$.* [16] *For any $\delta \in (0, 1)$ and $N \in \mathbb{N}^+$, consider a $4L$-layer* random *Transformer $\mathcal{T}_{large}$ with embedding dimension $m_{\mathrm{large}} = O(m \log(Lm/\delta))$, attention dimension $m_{\mathrm{large},a} = O(m_a L \log \frac{m_a m L N}{\epsilon\delta})$, and projection function $\mathrm{g}_{\mathrm{large}}$ as 4-layer ReLU MLP with width $w_{\mathrm{large}} = O(\max\{m, w\} L \log \frac{wmLN}{\epsilon\delta})$.*

Assume that $\|W\|_2 \leq 1$ for every weight matrix W in $\mathcal{T}$, and suppose the weights are randomly sampled as $W_{i,j} \sim U(-1, 1)$ for every $W \in \mathcal{T}_{large}$. Then, with probability $1 - \delta$ over the randomness of $\mathcal{T}_{large}$, there exists a nonstructural pruning (Definition 2.2.4) of $\mathcal{T}_{large}$, denoted as $\tilde{\mathcal{T}}_{large}$, which ϵ-approximates $\mathcal{T}$ with respect to $\|\cdot\|_{1,2}$ for any input $X \in \mathbb{R}^{m\times N}$ satisfying $\|\boldsymbol{X}\|_{1,2} \leq 1$. [17]

Proof sketch: connection to Lottery Tickets. Theorem 2.2.3 can be interpreted as a lottery ticket hypothesis (Frankle & Carbin, 2018; Malach et al., 2020) for randomly initialized Transformers, which can be of independent interest. The proof repeatedly uses an extension of Theorem 1 of Pensia et al. (2020), where it 1) first prunes the $(2l-1)$-th and $2l$-th layers of $\mathcal{T}_{\mathrm{large}}$ to approximate $\mathcal{T}^{(l)}$ for each $l \in [L]$ (Lemma 2.2.6), and 2) then prunes the remaining $2L + 1$ to $4L$-th layers of $\mathcal{T}_{\mathrm{large}}$ to approximate the identity function. The full proof is deferred to Section 2.2.5.5.

Noting that the layers used to approximate the identity can appear at arbitrary depth in $\mathcal{T}_{\mathrm{large}}$, a direct corollary of Theorem 2.2.3 is that one cannot distinguish between two functionally different Transformers by inspecting any single weight matrix only:

Corollary 2.2.3. *Let $\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_{large}$ follow the same definition and assumptions as $\mathcal{T}$ and $\mathcal{T}_{large}$ in Theorem 2.2.3. Pick any weight matrix W in $\mathcal{T}_{large}$, then with probability $1 - \delta$ over the randomness of $\mathcal{T}_{large}$, there exist two Transformers $\mathcal{T}_{Large,1}, \mathcal{T}_{Large,2}$ pruned from $\mathcal{T}_{large}$, such that $\mathcal{T}_{Large,i}$ ϵ-approximate $\mathcal{T}_i$, $\forall i \in \{1, 2\}$, and $\mathcal{T}_{Large,1}$, $\mathcal{T}_{Large,2}$ coincide on the pruned versions of W.*

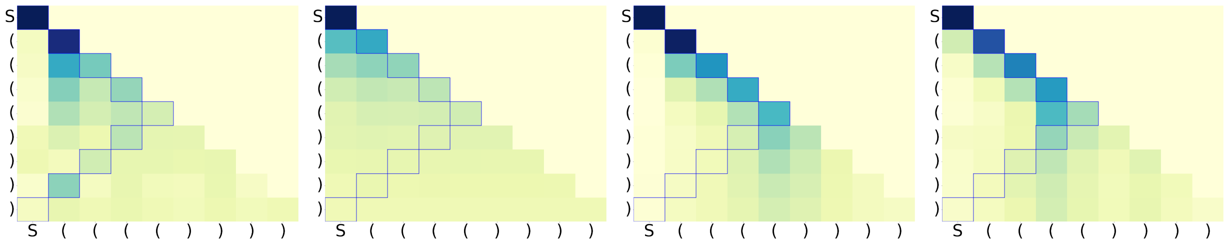
Hence, one should be cautious when using methods based solely on individual components to interpret the overall function of a Transformer.

2.2.3 Experiments

Our theory in Section 2.2.2 proves the existence of abundant *non-stack-like* attention patterns, all of which suffice for (near-)optimal generalization on Dyck. However, could it be that stack-like solutions are more frequently discovered empirically, due to potential *implicit biases* in the architecture and the training procedure? In this section, we show there is no evidence for such implicit bias in standard training (Section 2.2.3.2). Additionally, we propose a regularization term based on the balance condition (Theorem 2.2.1), which leads to better length generalization (Section 2.2.3.3).

[16] For notational convenience, we assume all layers share the same dimensions and projection functions. The proof can be trivially extended to cases where the dimensions and projection functions are different.

[17] Here the input and output dimension of $\tilde{\mathcal{T}}_{\mathrm{large}}$ is actually m_{large} which is larger than m; additional dimensions are padded with zeroes. The norm constraint can be easily extended to an arbitrary constant.



(a) Embedding 2.55, run 1 (b) Embedding 2.55, run 2 (c) Embedding 2.57 (d) Embedding 2.56

Figure 2.12: **Second-layer attention patterns of two-layer Transformers with a minimal first layer**: (a), (b) are based on embedding 2.55 with different learning rates, where the attention patterns show much variance as Theorem 2.2.1 predicts. (c), (d) are based on embedding 2.57 and 2.56. Different embedding functions lead to diverse attention patterns, most of which are not stack-like.

2.2.3.1 Training Details

For Figure 2.11, we train 2-layer standard GPT on $\mathsf{Dyck}_{2,4}$ with sequence length no longer than 28. For (a), we train with hidden dimension and network width 200 and learning rate 3e-4. For $(b), (c), (d)$, we train with hidden dimension and FFN width 50 and learning rate 3e-3.

For Figure 2.12, for (a), we train 1-layer transformer without residual link, FFN and the final LayerNorm before the linear head. The hidden dimensions and FFN widths are fixed as 500. For (a), we train the network with learning rate 1e-2 and for $(b), (c), (d)$ we train the network with learning rate 3e-3.

2.2.3.2 Different Attention Patterns Can Be Learned To Generate Dyck

We empirically verify our theoretical findings that Dyck solutions can give rise to a variety of attention patterns, by evaluating the accuracy of predicting the last bracket of a prefix (Equation 2.38) given the rest of the prefix. We only consider prefixes ending with a closing bracket, so that there exists a unique correct closing bracket which a correct parser should be able to determine. The experiments in this section are based on Transformers with 2 layers and 1 head, hidden dimension 50 and embedding dimension 50, trained using Adam. Additional results for three-layer Transformers are provided in Section 2.2.3.5. The training data consists of valid $\mathsf{Dyck}_{2,4}$ sequences of length less than 28 generated with $q = 0.5$. When tested in-distribution, all models are able to achieve $\geq 97\%$ accuracy.

Variation in attention patterns First, as a response to (Q1), we observe that attention patterns of Transformers trained on Dyck are not always stack-like (Figure 2.11). In fact, the attention patterns differ even across different random initialization. Moreover, while Theorem 2.2.1 implies that position encoding is not necessary for a Transformer to generate Dyck, [18] adding the position encoding [19] does affect the attention patterns (Figures 2.11c and 2.11d).

Specifically, for 2-layer Transformers with a minimal first layer, we experiment with three different types of embeddings $\boldsymbol{e}$: let $\boldsymbol{o}_t$ denote the one-hot embedding where $\boldsymbol{o}_t[t] = 1$,

$$\boldsymbol{e}(\tau_{t,d}) = \boldsymbol{o}_{(t-1)D+d} \in \mathbb{R}^{2kD}, \tag{2.55}$$

$$\boldsymbol{e}(\tau_{t,d}) = \boldsymbol{o}_t \oplus \boldsymbol{o}_d \in \mathbb{R}^{2k+D}, \tag{2.56}$$

$$\boldsymbol{e}(\tau_{t,d}) = \boldsymbol{o}_t \oplus [\cos(\theta_d), \sin(\theta_d)] \in \mathbb{R}^{2k+2}, \theta_d = \arctan(d/(D+2-d)), \tag{2.57}$$

where $\oplus$ denotes vector concatenation. Equation (2.55) is the standard one-hot embedding for $\tau_{t,d}$; Equation (2.56) is the concatenation of one-hot embedding of types and depths. Finally, Equation (2.57) is the

[18] This is verified empirically, as Transformers with no positional encoding achieve $\geq 97\%$ accuracy.

[19] We use the linear positional encoding following Yao et al. (2021): for the i_{th} position, the encoding is defined to be $\boldsymbol{e}_p(i) := i/T_{\max}$ for some $T_{\max}$.

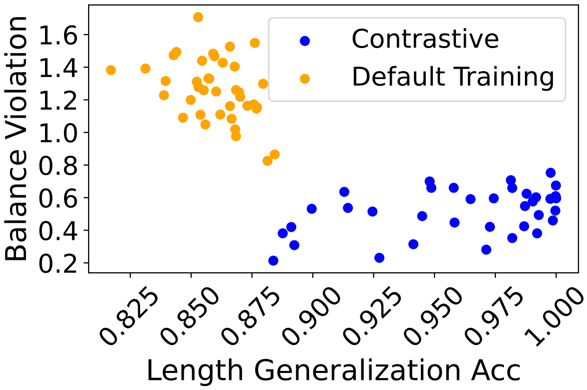


Figure 2.13: **Relationship Between Balance Violation and Length Generalization.** Accuracy from Transformers with minimal first layer with embedding 2.55, using both standard training and contrastive regularization (Equation (2.58)). Standard training leas to high balance violations which negatively correlate with length generalization performance. Contrastive regularization helps reduce the balance violation and improve the length generalization performance.

embedding constructed in Yao et al. (2021). As shown in Figure 2.12, the attention patterns learned by Transformers exhibit large variance between different choices of architectures and learning rates, and most learned attention patterns are not stack-like.

Quantifying the variation We now quantify the variation in attention by comparing across multiple random initializations. We define the *attention variation* between two attention patterns A_1, A_2 as $\text{Variation}(A_1, A_2) = \|A_1 - A_2\|_F^2$, for $A_1, A_2 \in \mathbb{R}^{N \times N}$ over an length-N input sequence. We report the *average attention variation* of each architecture based on 40 random initializations.

On the prefix [[[[]]]](((()))) [20], we observe that for standard two layer training, the average attention variation is 2.20 with linear position embedding, and is 2.27 without position embedding. Both numbers are close to the random baseline value of 2.85 [21], showing that the attention head learned by different initializations indeed tend to be very different. We also experiment with Transformer with a minimal first layer and the embedding in Equation (2.55), where the average variation is reduced to 0.24. We hypothesize that the structural constraints in this setting provide sufficiently strong inductive bias that limit the variation.

2.2.3.3 Guiding The Transformer To Learn Balanced Attention

In our experiments, we observe that although models learned via standard training that can generalize well in distribution, the *length generalization* performance is far from optimal. This implies that the models do not correctly identify the parsing algorithm for Dyck when learning from finite samples. A natural question is: can we guide Transformers towards correct algorithms, as evidenced by improved generalization performance on longer Dyck sequences?

In the following, we measure length generalization performance by the model accuracy on valid Dyck prefixes with length randomly sampled from 400 to 500, which corresponds to around 16 times the length of the training sequences. Inspired by results in Section 2.2.2, we propose a regularization term to encourage more balanced attentions, which leads to better length generalization.

Regularizing for balance violation improves length generalization accuracy We denote the *balance violation* of a Transformer as $\beta := \mathbb{E}_{d,d',i,j}\left[S_{d,d',i,j}/P_{d,j}\right]$ for S, P defined in Equations (2.51) and (2.53). Theorem 2.2.1 predicts that for models with a minimal first layer, perfect length generalization requires β to be zero. Inspired by this observation, we design a contrastive training objective to reduce the balance violation, which ideally would lead to improved length generalization. Specifically, let p_r denote a prefix of r nested pairs of brackets of for $r \sim U([D])$, and let $\mathcal{T}(s \mid p_r \oplus s)$ denote the logits for s when $\mathcal{T}$ takes as input the concatenation of p_r and s. We define the *contrastive regularization term* $R_{\text{contrastive}}(s)$ as the

[20] This prefix contains brackets of all types and depths. Results with different prefixes are provided in Section 2.2.3.5.

[21] The random baseline is calculated by generating purely random attention patterns (from the simplex, i.e. random square matrices s.t. each row sums up to 1) and calculate the average attention variation between them.

mean squared error between the logits of $\mathcal{T}(s)$ and $\mathcal{T}(s \mid p_r \oplus s)$, taking expectation over r and p_r:

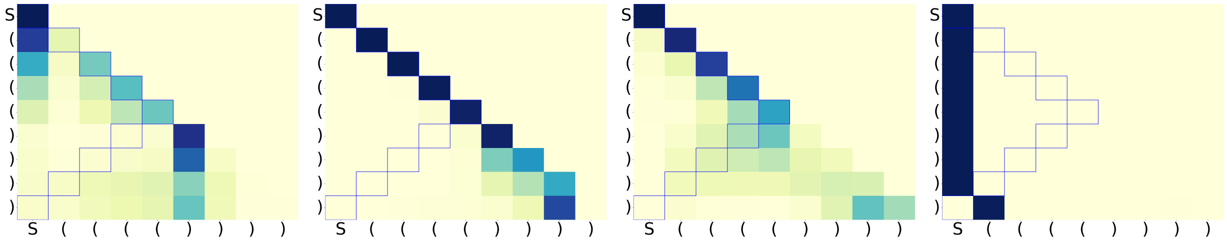
$$\mathbb{E}_{r\sim U([D]),p_r}\left[\|\mathcal{T}(s \mid p_r \oplus s) - \mathcal{T}(s)\|_F^2\right]. \tag{2.58}$$

Following the same intuition as in the proof of Theorem 2.2.1, if the model can perfectly length-generalize, then the contrastive loss will be zero. Models trained with contrastive loss show reduced balance violation as well as improved length generalization performance, as shown in Figure 2.13.

2.2.3.4 Results on Dyck prefixes

In the above experimental results, we perform experiments on *complete* Dyck sequences. In this section, we present additional experiments on Dyck *prefixes* $\mathsf{Dyck}_{2,4,28}$. Note that emphcomplete Dyck is a special case of Dyck *prefixes*.

Attention Patterns We first perform experiments on attention patterns. The qualitative results are shown in Figures 2.14 and 2.16. We can observe that the attention patterns are still diverse and do not commonly show stack-like patterns. We also calculate the *attention variation* [22], and find that the attention variation is 0.34, based on 30 models with a minimal first layer and different random seeds. In contrast, for models with a standard first layer and without position encodings, the attention variation is surprisingly high, reaching 14.51. The high value is caused by the large distance between attention patterns like Figure 2.14 (c) and (d); that is, between patterns that attend more to the current positions, and patterns that attend more heavily to the initial position. The difference is even increased when we consider longer sequence (Figure 2.15). Similarly, the variation is also high for models with linear position embedding, reaching 11.92. This shows that the attention patterns are still diverse and do not commonly show stack-like patterns.



(a) With Position Embedding

(b) With Position Embedding

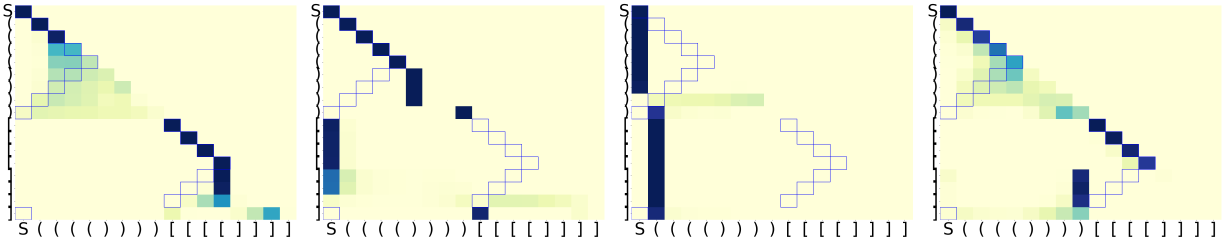
(c) Without Position Embedding

(d) Without Position Embedding

Figure 2.14: **Second-layer attention patterns of two-layer Transformers on Dyck Prefix**: Models for (a),(b) are under the same setup but different random seeds; similarly for (c),(d). All models reach $\geq 97\%$ accuracy (defined in Section 2.2.3.2). In the heatmap, darker color indicates larger value. As we can observe, the attention patterns still show much variance.

Balanced Violations We also test the relationship with the balance violation with length generalization on Dyck prefixes, similar to Figure 2.13. We observe that although the negative correlation is not presented as in the case of Dyck sequences, contrastive regularization still helps reduce the balance violation and significantly improve the length generalization performance. This shows that for Dyck prefixes, while the balance violation may not be predictive of the length generalization performance, it is still possible to reduce the balance violation and improve the length generalization performance. The results are shown in Figure 2.17.

[22] Recall from Section 2.2.3.2 that the attention variation between two attention patterns $A_1, A_2 \in \mathbb{R}^{N\times N}$ is defined as $\text{Variation}(A_1, A_2) = \|A_1 - A_2\|_F^2$.



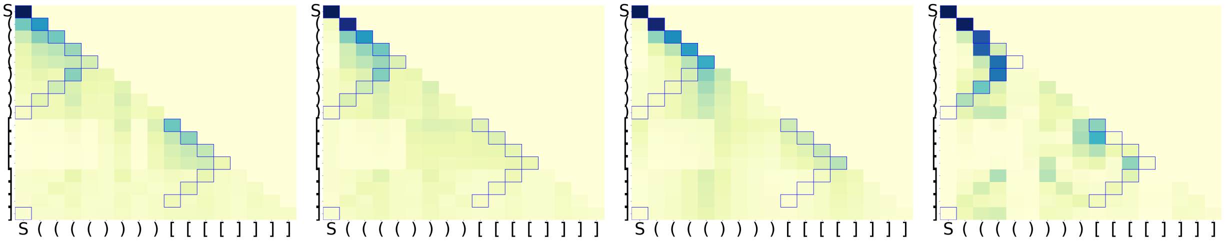
(a) With Position Embedding Run 1

(b) With Position Embedding Run 2

(c) Without Position Embedding Run 1

(d) Without Position Embedding Run 2

Figure 2.15: **Second-layer attention patterns of two-layer Transformers on Longer Dyck Prefix**: Models for (a),(b) are under the same setup but different random seeds. All models reach $\geq 97\%$ accuracy (defined in Section 2.2.3.2). In the heatmap, darker color indicates larger value.



(a) Embedding 2.55, run 1

(b) Embedding 2.55, run 2

(c) Embedding 2.57

(d) Embedding 2.56

Figure 2.16: **Second-layer attention patterns of two-layer Transformers with a minimal first layer**: (a), (b) are based on embedding 2.55 with different random seeds. (c), (d) are based on embedding 2.57 and 2.56. Different embedding functions lead to diverse attention patterns, most of which are not stack-like.

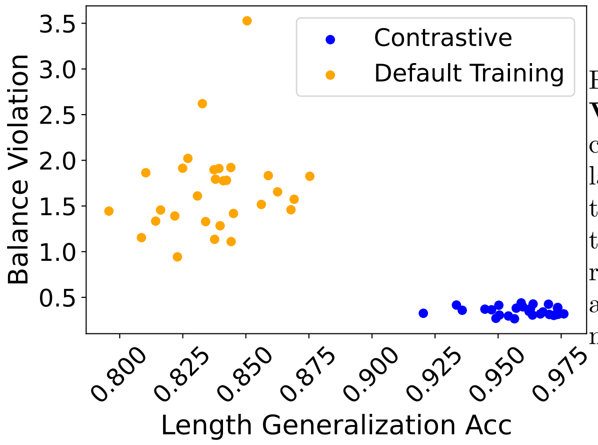


Figure 2.17: **Relationship Between Balance Violation and Length Generalization.** Accuracy from Transformers with minimal first layer with embedding 2.55, using both standard training and contrastive regularization (Equation (2.58)). We again observe that contrastive regularization helps reduce the balance violation and improve the length generalization performance.

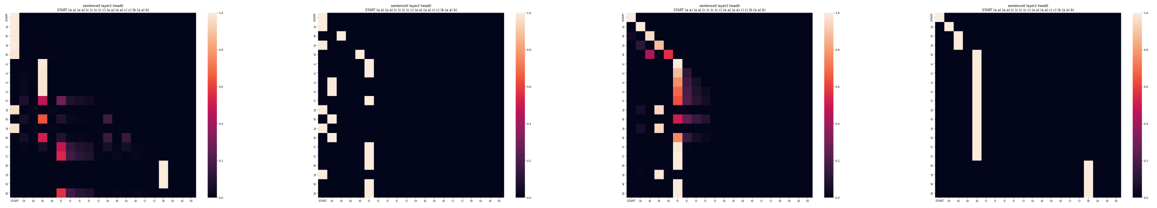
2.2.3.5 Additional variants of Dyck languages and architectures

We include more experiments on the attention variation of different Dyck languages and architectures. The results are summarized in Table 2.6.

#types k	Grammar depth m	#Layers l	Layer 1	Layer 2	Layer 3
2	4	2	$0.047_{(0.006)}$	$7.721_{(0.908)}$	
2	4	3	$0.070_{(0.013)}$	$5.072_{(0.645)}$	$24.063_{(1.166)}$
2	8	2	$0.087_{(0.012)}$	$7.583_{(0.961)}$	
2	8	3	$0.059_{(0.011)}$	$5.560_{(0.714)}$	$23.590_{(0.829)}$
3	4	2	$0.182_{(0.024)}$	$9.313_{(0.815)}$	
3	4	3	$0.225_{(0.032)}$	$8.426_{(0.877)}$	$25.749_{(0.897)}$
3	8	2	$0.178_{(0.028)}$	$7.000_{(0.884)}$	
3	8	3	$0.154_{(0.036)}$	$6.280_{(0.711)}$	$25.451_{(0.871)}$

Table 2.6: **Extended attention variation.** "Layer i" shows the mean (and standard deviation) of the attention variation on layer i, calculated on 40 sentences. The embedding width and FFN width are fixed as 50 in the experiments. We train using sentences from $\mathsf{Dyck}_{k,m}$ of length less than 28 and test the variation on 40 randomly sampled sentences with length 19 (the sampled sentence is fixed across different architectures). The random attention variation baseline here is 3.33. The numbers in this table are different from previous discussion, since the results here are from a slightly different architecture than the standard GPT-2 architecture: a residue link is appended after the LayerNorm to match our theory better. The models are trained to convergence and have in-distribution accuracy higher than 97%.

Attention pattern visualization for three-layer experiments. The first-layer attention is close to uniform, while the higher-layer attention shows no clear patterns.



(a) *seed* = 0 (b) *seed* = 1 (c) *seed* = 2 (d) *seed* = 3

Figure 2.18: **Third-layer attention.** The test sentence is fixed and the attention patterns learned by different 3-layer models with the same architectures on the same dataset show large variation visually.

2.2.4 Discussion: are interpretable attention patterns useful?

Our results Section 2.2.2 and Section 2.2.3.2 demonstrate that Transformers are sufficiently expressive that a (near-)optimal loss on Dyck languages can be achieved by a variety of attention patterns, many of which may not be interpretable.

However, multiple prior works have shown that for multi-layer multi-head Transformers trained on natural language datasets, it is often possible to locate attention heads that produce interpretable attention patterns (Vig & Belinkov, 2019; Htut et al., 2019; Sun & Marasović, 2021). Hence, it is also illustrative to consider the *"converse question"* of (Q1): when some attention heads do learn to produce attention patterns that suggest intuitive interpretations, what benefits can they bring?

We discuss this through two perspectives:

- **Reliability of interpretation:** Is the Transformer necessarily implementing a solution consistent with such interpretation based on the attention patterns? (Section 2.2.4.1)

- **Usefulness for task performance:** Are those interpretable attention heads more important for the task than other uninterpretable attention heads? (Section 2.2.4.2)

We present preliminary analysis on these questions, and motivate future works on the interpretability of attention patterns using rigorous theoretical analysis and carefully designed experiments.

2.2.4.1 Can interpretable attention patterns be misleading?

We show through a simple argument that interpretations based on attention patterns can sometimes be misleading, as we formalize in the following proposition:

Proposition 2.2.1. *Consider an L-layer Transformer $\mathcal{T}$ (Equation (2.46)). For any $W_K^{(l)}, W_Q^{(l)} \in \mathbb{R}^{m_a \times m}$ ($l \in [L]$), there exist $W_{\text{Head}} \in \mathbb{R}^{2k \times w}$ and $b_{\text{Head}} \in \mathbb{R}^{2k}$ such that $\mathcal{T}(\mathcal{Z}) = 0, \forall \mathcal{Z}$.*

While its proof is trivial (simply setting $W_{\text{Head}} = 0$ and $b_{\text{Head}} = 0$ suffices), Proposition 2.2.1 implies that the solution represented by the Transformer could possibly be independent of the attention patterns in all the layers (1 through l). Hence, it could be misleading to interpret Transformer solutions solely based on these attention patterns.

Empirically, Transformers trained on Dyck indeed sometimes produce misleading attention patterns.

We present one representative example in Figure 2.19, and Figure 2.20, in which *all interpretable attention patterns are misleading.*

We also present additional results in Figure 2.21, in which *some interpretable attention patterns are misleading, and some are not.*

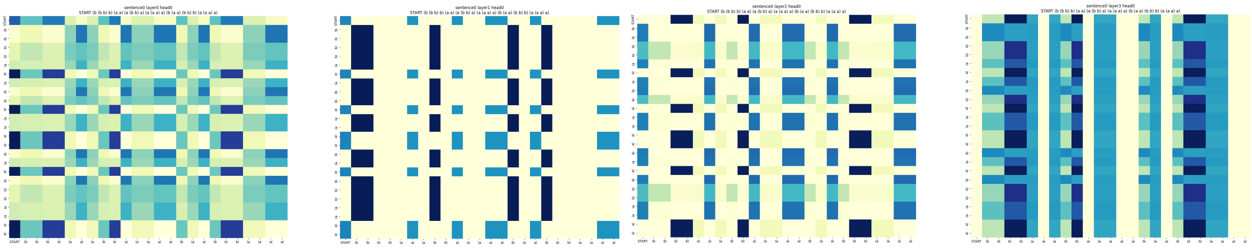


Figure 2.19: **Even interpretable attention patterns can be misleading**: For a 4-layer Transformer trained on Dyck with the *copying* task (with > 96% validation accuracy), i.e. the output should be exactly the same as the input, the attention patterns in some layers seem interpretable: (layer 2) attending to bracket type a) or (b; (layer 3) attending to closing bracketss; (layer 4) neve attending to bracket type a); However, none of them are informative of the copying task. This is possible because Transformers can use the residual connections (or weights MLPs or the value matrices) to solve copying, bypassing the need of using attention.

Similar message has been conveyed in prior works Bolukbasi et al. (2021), and future works may aim to achieve the *faithfulness*, *completeness*, and *minimality* conditions in Wang et al. (2023).

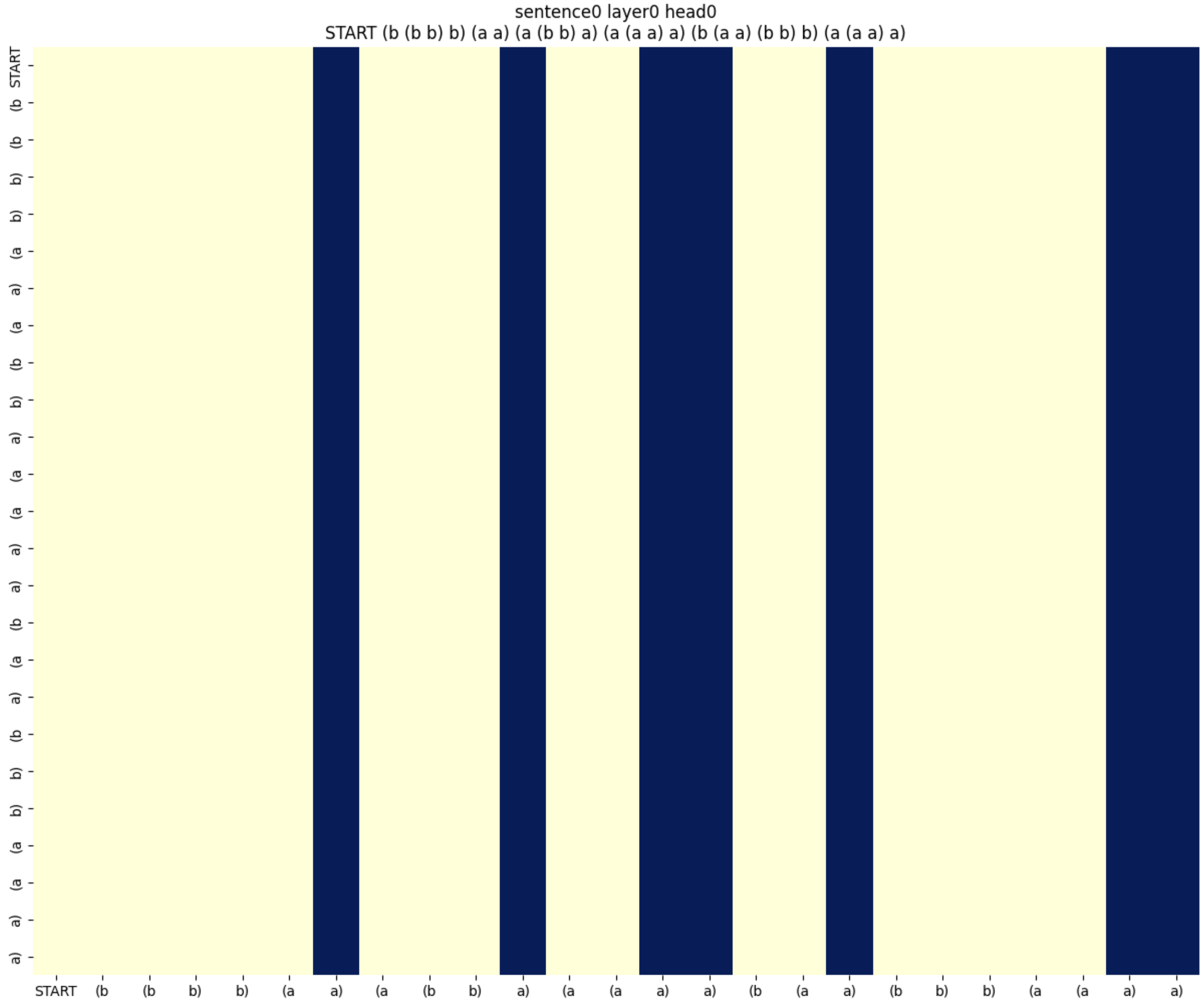
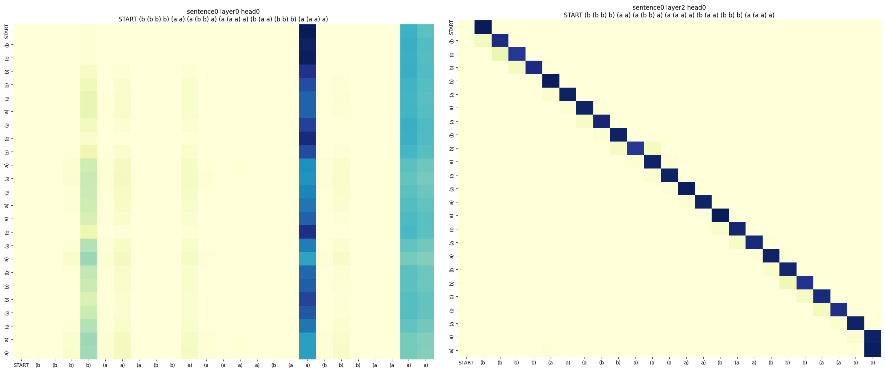


Figure 2.20: **Even interpretable attention patterns can be misleading**: For a 1-layer Transformer trained on Dyck with the *copying* task (with > 90% validation accuracy), i.e. the output should be exactly the same as the input, the attention pattern seems to be attending to closing brackets only, but that is not informative of the copying task.



(a) layer 1 of 4

(b) layer 3 of 4

Figure 2.21: **Even interpretable attention patterns can be misleading**: For a 4-layer Transformer trained on Dyck with the *copying* task (with $> 96\%$ validation accuracy), i.e. the output should be exactly the same as the input, both types of attention patterns are common: (a) attending to closing bracketss, which is uninformative of the copying task; (b) attending to the current position, which solves the copying task.

2.2.4.2 Are attention heads with interpretable patterns more important?

Kovaleva et al. (2019) observes that, when the "importance" of an attention head is defined as the performance drop the model suffers when the head is disabled, then for most tasks they test, the most important attention head in each layer *does not* tend to be interpretable.

However, experiments by Voita et al. (2019) led to a seemingly contradictory observation: when attention heads are systematically pruned by finetuning the Transformer with a relaxation of L_0-penalty (i.e. encouraging the number of remaining attention heads to be small), most remaining attention heads that survive the pruning can be associated with certain functionalities such as positional, syntactic, or attending to rare tokens.

These works seem to bring mixed conclusions to our question: are interpretable attention heads more important for a task than uninterpretable ones? We interpret these results by conjecturing that the definition of "importance" (reflected in their experimental design) plays a crucial role:

- When the importance of an attention head is defined *treating all other attention heads as fixed*, motivating experiments that prune/disable certain heads while keeping other heads unchanged (Michel et al., 2019; Kovaleva et al., 2019), the conclusion may be mostly pessimistic: mostly no strong connection between interpretability and importance.

- On the other hand, when the importance of an attention head is defined *allowing all other attention heads to adapt to its change*, motivating experiments that jointly optimize all attention heads while penalizing the number of heads (Voita et al., 2019), the conclusion may be more optimistic: the heads obtained as a result of this optimization tend to be interpretable.

We think the following trade-offs apply:

- On one hand, the latter setting is more practical, since Transformers are typically not trained to explicitly ensure that the model performs well when a single attention head is individually disabled; rather, it would be more intuitive to think of a group of attention heads as jointly representing some transformation, so when one head is disabled, other heads should be fine-tuned to adapt to the change.

- On the other hand, when all other heads change too much during such fine-tuning, the resulting set of attention heads no longer admit an unambiguous one-to-one map with the original set of (unpruned) attention heads. As a result, the interpretability and importance obtained from the set of pruned heads do not necessarily imply those properties of the original heads.

A comprehensive study of this question involves multi-head extensions of our theoretical results (Section 2.2.2), and carefully-designed experiments that take the above-mentioned trade-offs into consideration. We think these directions are interesting future work.

2.2.5 Omitted Proofs in Section 2.2.2

2.2.5.1 Proof of Theorem 2.2.1

The key step was outlined in Section 2.2.2. We will restate the proof rigorously here.

Theorem 2.2.1 (Perfect Balance)**.** *Consider a two-layer Transformer $\mathcal{T}$ (Equation (2.46)) with a minimal first layer (Assumption 2.2.1) and $C = 0$ (Equation (2.43)). Let $\mathcal{O}$ denote the* optimal prediction *scenario, that is, when the first layer embeddings $\{\boldsymbol{e}(\tau_{i,d})\}_{d\in[D],i\in[2k]}$ (Definition 2.2.2) and second layer parameters $\theta^{(2)}$ satisfy*

$$\theta := \{\boldsymbol{e}(\tau_{i,d})\}_{d\in[D],i\in[2k]}, \theta^{(2)}\} = \arg\min_{\tilde{\theta}} \mathcal{L}(\tilde{\theta}; \mathcal{D}_{\mathsf{Dyck}}), \forall N,$$

where the objective $\mathcal{L}$ is defined in Equation (2.40). Then,

- *Equation (2.47) is a necessary condition of $\mathcal{O}$, if $W_V^{(2)}$ satisfies $\mathcal{P}_\perp W_V^{(2)} \boldsymbol{e}(\tau_{t,d}) \neq 0, \forall t \in [k], d \in [D]$.*

- *Equation (2.47) is a sufficient condition of $\mathcal{O}$, for a construction in which the set of $2k+1$ encodings $\{\boldsymbol{e}(\tau_{2i-1,d}), \boldsymbol{e}(\tau_{2i,d})\}_{i\in[k]} \cup \{\boldsymbol{e}(t_{\mathcal{S}})\}$ are linearly independent for any $d \in [D]$ and the projection function $\mathrm{g}^{(2)}$ is a 6-layer MLP* [23] *with $O(k^2D^2)$ width.*

Proof. We prove the **sufficiency of the balanced condition** below; the proof for the *necessity* has been given in Section 2.2.2.

We will denote the dimension of $\boldsymbol{e}(\tau_{t,d})$ as m.

For any $i \in [k], d' \in [D]$, by Equation (2.47), we can assume that there exists $a_{i,d'} \in \mathbb{R}$ such that for all $j \in [k]$, $d \in [D]$, it holds that,

$$a_{i,d'} \triangleq \left(\boldsymbol{e}\left(\tau_{2i-1,d'}\right) - \boldsymbol{e}\left(\tau_{2i,d'-1}\right)\right)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\left(\tau_{2j,d}\right). \tag{2.59}$$

We will first define the possible index sets of $\tau_{t,d}$ as $\mathcal{I} = \{(2t,d) \mid t \in [k], 0\ \leq d \leq D-1\} \cup \{(2t-1,d) \mid t \in [k], 1 \leq d \leq D\}$, and we will define the rank of (t,d) as

$$r(t,d) \triangleq \#\{(t_1,d_1) \mid t_1 < t \text{ or } t_1 = t, d_1 \leq d, (t_1,d_1) \in \mathcal{I}\} \tag{2.60}$$

Then it is clear that $r(t,d)$ is a one-to-one mapping from $\mathcal{I}$ to $[2kD]$. We will then define the collection of all $\boldsymbol{e}(\tau_{t,d})$ as $\boldsymbol{E}$, satisfying that $\boldsymbol{E}_{:,r(t,d)} = \boldsymbol{e}(\tau_{t,d}), \boldsymbol{E}_{:,2kD+1} = \boldsymbol{e}(t_{\mathcal{S}})$.

Because $\boldsymbol{e}(\tau_{t,d})$ are linearly independent, for any $(i,d) \neq (j,d') \in \mathcal{I}$, it holds that $\boldsymbol{e}(\tau_{i,d}) - \boldsymbol{e}(\tau_{j,d'}) \neq 0$. Then based on Lemma 2.2.16, there exists a set of orthonormal vectors $\{\mathrm{b}_i\}_{i\in[m-2]}$, such that for any $(i,d),(j,d') \in \mathcal{I}$, it holds that

$$\sum_{i=1}^{m-2} \mathrm{b}_i \mathrm{b}_i^\top \left(\boldsymbol{e}(\tau_{i,d}) - \boldsymbol{e}(\tau_{j,d'})\right) \neq \left(\boldsymbol{e}(\tau_{i,d}) - \boldsymbol{e}(\tau_{j,d'})\right) \tag{2.61}$$

$$\mathrm{b}_i^\top 1^m = 0 \tag{2.62}$$

We will further construct the matrix $\boldsymbol{O}$ as [24]

$$\begin{aligned} \boldsymbol{O}_{:,r(2t,d-1)} &= -\exp(a_{t,d})\mathrm{b}_{tD+d}, \\ \boldsymbol{O}_{:,r(2t-1,d)} &= \mathrm{b}_{tD+d}. \\ \boldsymbol{O}_{:,2kD+1} &= 0. \end{aligned} \tag{2.63}$$

for $t \in [k], d \in [D]$.

We can then choose $W_V^{(2)} \in \mathbb{R}^{m\times m}$ such that

$$W_V^{(2)}\boldsymbol{E} = \boldsymbol{O} \tag{2.64}$$

Such $W_V^{(2)}$ is guaranteed to exist, because $\boldsymbol{E}$ is of full column rank by the linear independence assumption.

Now based on this construction, we will show that the last column of unnormalized attention output (Equation (2.45)) depends only on the sequence of unmatched brackets when the last token is a closed bracket with depth d greater than or equal to 1. [25]

For any valid Dyck prefix p of length n ending with a closed bracket $\tau_{2j,d}$ satisfying $d \geq 1$, suppose the list of unmatched open brackets in p is $[\tau_{2j_1-1,1}, \tau_{2j_2-1,2}, \ldots, \tau_{2j_d-1,d}]$. Then, the remaining tokens in p are

[23] In the construction, we first use 4 layers to convert the input of the projection function to a triplet indicating the type and depth of the last token and the type of the last unmatched bracket when the last token is a closed bracket. We then use another 2 layers to predict the next token probability based on the triplet. This construction is likely improvable.

[24] Recall the definition of r in Equation (2.60). Comparing $\boldsymbol{O}_{:,r(2t,d-1)}$ and $\boldsymbol{O}_{:,r(2t-1,d)}$: the idea is that a pair of matched brackets are represented by the same direction (i.e. the direction along b_{tD+d}), just with different norms.

[25] When depth $d = 0$, all brackets are matched, the groundtruth next-token distribution is the prior distribution over the open brackets. Because in Equation (2.47) $d_1, d_2 \geq 1$, we handle the depth $d = 0$ case separately in Case 2 "t is even, $d = 0$" towards the end of this proof. In the following, we focus on cases with depth $d \geq 1$.

pairs of matching brackets. Denote them by $\tau_{2t_k-1,d_k}, \tau_{2t_k,d_k-1}$ for $k \in [K]$. Then the input of the second layer of Transformer X, up to a permutation is

$$XP = [\boldsymbol{e}(\tau_{2t_1-1,d_1}), \boldsymbol{e}(\tau_{2t_1,d_1-1}), \dots, \boldsymbol{e}(\tau_{2t_K-1,d_K}), \boldsymbol{e}(\tau_{2t_K,d_K-1}), \boldsymbol{e}(\tau_{2j_1-1,1}), \dots \boldsymbol{e}(\tau_{2j_d-1,d}), \boldsymbol{e}(t_{\mathcal{S}})].$$

We will focus on the last column of the unnormalized attention output

$$\begin{aligned}
\tilde{a}_2(X;\theta^{(2)})_{:,n+1} &= \mathcal{P}_\perp \Big[W_V^{(2)} X \cdot \tilde{\sigma}\Big(\mathcal{C} \cdot (W_K^{(2)}X)^\top (W_Q^{(2)}X)\Big)\Big]_{:,n+1} \\
&= \sum_{s=1}^{n+1} \mathcal{P}_\perp (W_V^{(2)}X)_{:,s} \Big[\tilde{\sigma}\Big(\mathcal{C} \cdot (W_K^{(2)}X)^\top (W_Q^{(2)}X)\Big)\Big]_{s,n+1} \\
&= \sum_{s=1}^{n+1} \mathcal{P}_\perp (W_V^{(2)}X)_{:,s} \exp\Big(\Big((W_K^{(2)}X)^\top (W_Q^{(2)}X)\Big)_{s,n+1}\Big) \\
&= \sum_{s=1}^{n+1} \mathcal{P}_\perp (W_V^{(2)}X)_{:,s} \exp\Big((W_K^{(2)}X)_{:,s}^\top (W_Q^{(2)}X)_{:,n+1}\Big) \\
&= \sum_{k=1}^{K} \left(u(\tau_{2t_k,d_k-1}, \tau_{2j,d}) + u(\tau_{2t_k-1,d_k}, \tau_{2j,d})\right) + \sum_{s=1}^{d} u(\tau_{2j_s-1,s}, \tau_{2j,d})
\end{aligned} \tag{2.65}$$

in which the last line is by definition of $u(\cdot,\cdot)$ in Equation (2.48).

For any indices s, j_s, j, d, we can simplify the expression for $u(\tau_{2j_s-1,s}, \tau_{2j,d})$ by observing that

$$\begin{aligned}
u(\tau_{2j_s-1,s}, \tau_{2j,d}) &= \mathcal{P}_\perp \exp\Big(\boldsymbol{e}\big(\tau_{2j_s-1,s}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\Big)\, W_V^{(2)} \boldsymbol{e}\big(\tau_{2j_s-1,s}\big) \text{ by Eq 2.48} \\
&= \mathcal{P}_\perp \exp\Big(\boldsymbol{e}\big(\tau_{2j_s-1,s}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\Big)\, \boldsymbol{O}_{:,r(2j_s-1,s)} \quad \text{by Eq 2.64} \\
&= \mathcal{P}_\perp \exp\Big(\boldsymbol{e}\big(\tau_{2j_s-1,s}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\Big)\, \mathrm{b}_{j_s D+s} \quad \text{by Equation (2.63)} \\
&= \exp\Big(\boldsymbol{e}\big(\tau_{2j_s-1,s}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\Big)\, \mathrm{b}_{j_s D+s} \quad \text{by Equation (2.62).}
\end{aligned} \tag{2.66}$$

Likewise by Equation (2.48), Equation (2.64), Equation (2.63), Equation (2.62)

$$u(\tau_{2j_s,s-1}, \tau_{2j,d}) = -\exp\Big(\boldsymbol{e}\big(\tau_{2j_s,s-1}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\Big) \exp(a_{j_s,s}) \mathrm{b}_{j_s D+s} \tag{2.67}$$

By Equation (2.66) and Equation (2.67),

$$\begin{aligned}
& u(\tau_{2t_k,d_k-1}, \tau_{2j,d}) + u(\tau_{2t_k-1,d_k}, \tau_{2j,d}) \\
&= \exp\Big(\boldsymbol{e}\big(\tau_{2t_k-1,d_k}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\Big)\, \mathrm{b}_{t_k D+d_k} \\
&\quad - \exp\Big(\boldsymbol{e}\big(\tau_{2t_k,d_k-1}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\Big) \exp(a_{t_k,d_k}) \mathrm{b}_{t_k D+d_k} \\
&= \big[\exp\Big(\boldsymbol{e}\big(\tau_{2t_k-1,d_k}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\Big) \\
&\quad - \exp\Big(\boldsymbol{e}\big(\tau_{2t_k,d_k-1}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big) + a_{t_k,d_k}\Big) \big] \mathrm{b}_{t_k D+d_k} \\
&= 0
\end{aligned} \tag{2.68}$$

in which the last line is because the terms inside $\big[\cdots\big]$ cancel each other, because by Equation (2.59)

$$\boldsymbol{e}\big(\tau_{2t_k-1,d_k}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big) = \boldsymbol{e}\big(\tau_{2t_k,d_k-1}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big) + a_{t_k,d_k}$$

Plugging Equation (2.68) and Equation (2.66) into Equation (2.65),

$$\begin{aligned}\tilde{a}_2(X;\theta^{(2)})_{:,n+1} &= \sum_{s=1}^{d} u(\tau_{2j_s-1,s},\tau_{2j,d}) \\ &= \sum_{s=1}^{d} \exp\left(\boldsymbol{e}\big(\tau_{2j_s-1,s}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\right) \mathrm{b}_{j_sD+s} \qquad (2.69)\end{aligned}$$

Therefore, $\tilde{a}_2(X;\theta^{(2)})_{:,n+1}$ lies in the span of $\{\mathrm{b}_{j_sD+s}\}_{s\in[d]}$. We will from now on assume

$$\langle \mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n}), \mathrm{b}_{j_sD+s}\rangle > M$$

for all possible choices of p ending with a closed bracket with grammar depth at least 1 for some constant $M\in(0,1)$. Here M exists because

$$\langle \mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n}), \mathrm{b}_{j_sD+s}\rangle = \frac{\exp\left(\boldsymbol{e}\big(\tau_{2j_s-1,s}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\right)}{\sqrt{\sum_{s'=1}^{d} \exp\left(2\boldsymbol{e}\big(\tau_{2j'_s-1,s}\big)^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}\big(\tau_{2j,d}\big)\right)}} > 0,$$

for all possible combination of $j_k, k\in[d]$ and s, and there are only finite number of such combinations.

Constructing the projection function $\mathrm{g}^{(2)}$ We will finally show there exists a 6-layer MLP $\mathrm{g}^{(2)}$ with width $O(D^2k^2)$, such that for any dyck prefix q with n being the length of q, X being the input of the second layer given q and $\mathbb{P}(p)$ being the groundtruth next-token probability vector given q [26], it holds that, $\mathrm{g}^{(2)}\left(\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1}) + X_{:,n+1}\right) = \mathbb{P}(q)$.

We will assume the last token of q is $\tau_{t,d}$. Suppose that $\mathrm{b}_{m-1}, \mathrm{b}_m$ is an orthonormal basis of the normal space of $\mathrm{span}\{\mathrm{b}_1,..,\mathrm{b}_{m-2}\}$, then we can first observe that for $U = \mathrm{b}_m\mathrm{b}_m^\top + \mathrm{b}_{m-1}\mathrm{b}_{m-1}^\top$, it holds that

$$U(\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1}) + X_{:,n+1}) = U\boldsymbol{e}(\tau_{t,d}).$$

is unique for every t,d. Then based on Lemma 2.2.15, there exists a 2-layer MLP with width $4kD$ that maps $U(\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1})+X_{:,n+1})$ to (t,d). This implies that there exists a 2-layer MLP with width $4kD$ that maps $\mathrm{LN}((\tilde{a}_2(X;\theta^{(2)})_{:,n}) + X_{:,n}$ to (t,d).

Further, let matrix $U' = \sum_{j=1}^{Dk} \boldsymbol{o}_j\mathrm{b}_j^\top$ where $\boldsymbol{o}_j$ is the Dk dimension one-hot vector with the $j-$th entries being 1. Then when t is an even number and $d\ge 1$, based on Equation (2.69) and the definition of M,

$$U'(\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1}) + X_{:,n+1})_{t'D+d'} \begin{cases} = 0, & \tau_{2t'-1,d'} \text{ is not an unmatched open brackets in } p. \\ > M, & \tau_{2t'-1,d'} \text{ is an unmatched open brackets in } p. \end{cases}$$

Then based on Lemma 2.2.18, there exists 2-layer MLP with width kD that operates on

$$\Big(U'(\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1}) + X_{:,n+1})_{t'D+d'}\Big)_{t'\in[k]}$$

for a fixed d' and output the nonzero index in it, if such index exists. Hence, we can choose the weight of the first and second layer of $\mathrm{g}^{(2)}$, such that the output of the second layer is $(t,d)\oplus x$, where $2x_{d'}-1$ is the type of the unmatched open brackets with grammar depth d' if t is an even number, $d\ge d'\ge 1$.

Now based on Lemma 2.2.17, we can choose the third and fourth layer of $\mathrm{g}^{(2)}$ to perform indexing and let the output of the fourth layer be (t,d,y), where $y = x_d$ when $d\ge 1$. [27] Notice that this triplet contains all the necessary information to infer $\mathbb{P}(q)$ because it uniquely determines the type of last unmatched open bracket,

[26] That is $\mathbb{P}(q)_t = \mathbb{P}(\text{The next token of } q \text{ has type } t)$

[27] When $d=0$, y does not matter since there is no unmatched open brackets.

1. If t is odd (i.e. the last bracket is open), and then the type of last unmatched open bracket is t.
2. If t is even and $d = 0$, then all the brackets is matched.
3. If t is even and $d \geq 1$, then the type of last unmatched bracket is y.

One may finally construct a 2-layer MLP f that maps (t, d, y) to the corresponding probability vector. As the input of g has bounded norm,

$$\|\mathrm{LN}(\tilde{a}_2(X; \theta^{(2)})_{:,n+1}) + X_{:,n+1}\|_2 \leq 1 + \max_{t,d} \|\boldsymbol{e}(\tau_{t,d})\|,$$

the output of the constructed 4 layers also has a bounded norm. Hence, we can assume there exists constant $M' > 1$, such that $y \leq M'$. Now we will discuss by the value of t,

1. t is odd, then one can neglect the third dimension and the correct probability is determined by d and can be represented by a width-$2D$ network based on Lemma 2.2.15.
2. t is even. When $d = 0$, one can construct a width-1 network mapping any y to the correct probability distribution as it is unique. When $d \geq 1$, one can construct a width-$2K$ network mapping $x_d \in [K]$ to the correct probability distribution based on Lemma 2.2.15. Then by Lemma 2.2.19, one can construct a width-$4KD$ network that maps (d, y) to the corresponding probability distribution.

Putting together and using Lemma 2.2.19 again, one can construct a width-$8K^2D$ network that maps (t, d, y) to the correct next token probability prediction. The proof is then completed.

□

2.2.5.2 Implication of our results to larger models

Recall that the main conclusion of our paper is that interpretability based on a single Transformer component (e.g. an attention pattern or an MLP block) can be unreliable, since the set of optimal solutions can give rise to a large set of attention patterns and pruned MLP weights. Section 2.2.2 has demonstrated this with simple two-layer Transformers. The simplicity of this architecture choice is intentional, since our theory on two-layer Transformers directly implies similar conclusions for larger models, as we discuss in this section.

Intuitively, when moving to more complex architectures, the set of solutions can only grow and complicate interpretability further, hence our main conclusion still stands. For example, even though Theorem 2.2.1 and Theorem 2.2.2 are stated for 2-layer Transformers only, the constructed solutions can be trivially extended to multiple layers by e.g. letting the higher layers perform the identity function, or removing Assumption 2.2.1 and allowing the model to flexibly use or ignore positional information. More precisely:

- For Transformers with greater width, our Theorem 2.2.1 applies directly, since the construction does not depend on the width.
- For Transformers with greater depth, it suffices to show that additional layers can perform the identity function. To this end, one can utilize the residue link in the Transformer layer and choose the value matrix to be zero and the FFN (with or without residue connection) to be identity. This construction is implicitly assuming LayerNorm will map zero vector to zero vector, which is true for the common PyTorch implementation and for our paper. Also, it is worth noting that this holds for both the architecture we considered in the paper and the standard GPT-2 architecture.

2.2.5.3 Proof of Corollary 2.2.1

Corollary 2.2.1. *There exists a 2-layer Transformer with uniform attention and no positional embedding (but with causal mask and a starting token* [28] *) that generates the Dyck language of arbitrary length.*

[28] Here the starting token is necessary because otherwise, the Transformer with uniform attention will have the same outputs for prefix p and prefix $p \oplus p$, in which $\oplus$ denotes concatenation, i.e. $p \oplus p$ means the same string p repeated twice.

Proof. We will first construct a uniform attention first layer that can generate the embedding in Equation (2.55). Suppose Z is the one-hot embeddings of a prefix p of length n, where each token of type t for $t \in [2k]$ is encoded as $\boldsymbol{o}_t$ and the starting token is encoded as $\boldsymbol{o}_{2k+1}$. Then it holds that

$$\left[Z\sigma\Big(\mathcal{C} \cdot (W_K^{(1)} Z)^\top (W_Q^{(1)} Z)\Big)\right]_{:,n+1} = \sum_{i=1}^{2k} \#\{\text{token of type } t \text{ in } p\}\boldsymbol{o}_t + \boldsymbol{o}_{2k+1}. \tag{2.70}$$

Then we can choose $W_V^{(1)}$ such that for $x \in \mathbb{R}^{2k+1}$,

$$\begin{aligned}(W_V^{(1)} x)_1 =& \sum_{i=1}^{k} x_{2i-1} - x_{2i},\\ (W_V^{(1)} x)_2 =& x_{2k+1},\\ (W_V^{(1)} x)_i =& 0, \forall i \geq 3.\end{aligned}$$

Hence it holds That

$$\left[W_V^{(1)} Z\sigma\Big(\mathcal{C} \cdot (W_K^{(1)} Z)^\top (W_Q^{(1)} Z)\Big)\right]_{:,n+1} = \#\{\text{depth of } p_n\}\boldsymbol{o}_1 + \boldsymbol{o}_2.$$

It is then easy to check $\mathrm{LN}\left(\left[W_V^{(1)} Z\sigma\Big(\mathcal{C} \cdot (W_K^{(1)} Z)^\top (W_Q^{(1)} Z)\Big)\right]_{:,n+1}\right) + Z_{:,n+1}$ is uniquely determined by the type and depth of p_n without repetition. Then by Lemma 2.2.15, there exists a 2-layer ReLU MLP with width $O(k^2 D^2)$ that can map $\mathrm{LN}\left(\left[W_V^{(1)} Z\sigma\Big(\mathcal{C} \cdot (W_K^{(1)} Z)^\top (W_Q^{(1)} Z)\Big)\right]_{:,n+1}\right) + Z_{:,n+1}$ to the embedding in Equation (2.55). It is then easy to see that the condition in Theorem 2.2.1 is satisfied as $W_K^{(2)} = W_Q^{(2)} = 0$. Hence the second layer can be constructed to let the Transformer to output the correct next token probability. □

2.2.5.4 Proof of Theorem 2.2.2

Theorem 2.2.2 (Approximate Balance)**.** *Consider a 2-layer Transformer $\mathcal{T}$ (Equation (2.46)) with a minimal first layer (Assumption 2.2.1) and a γ-Lipschitz $\mathrm{g}^{(2)}$ for $\gamma > 0$, trained on sequences of length N with the mean squared loss (Equation (2.41)).*

Suppose the loss is approximately optimal, precisely, the set of second-layer weights $\bar{\theta}_N^{(2)}$ satisfies

$$\mathcal{L}(\mathcal{T}[\bar{\theta}_N^{(2)}], \mathcal{D}_{\mathsf{Dyck}}) \leq (\frac{q(1-q)}{k^2})^N \epsilon$$

for every positive integer $N > 8D$ and sufficiently small $\epsilon > 0$. Then, there exists a constant $C_{\gamma,\epsilon,D}$, such that $\forall 0 \leq d' \leq D, 1 \leq d \leq D, i, j \in [k]$, it holds that

$$\|S_{d,d',i,j}[\bar{\theta}_N^{(2)}]\| \leq \frac{C_{\gamma,\epsilon,D}}{N} P_{d,j}[\bar{\theta}_N^{(2)}]. \tag{2.54}$$

Proof. The key idea is similar to the proof of necessity in Theorem 2.2.1. That is, we will construct two input sequences with different next-word distributions, and show that the approximate balance condition must hold so that inserting (a bounded number of) pairs of matching brackets does not collapse the two predicted distributions given by the Transformer.

Constructing the input sequences.

Let $\boldsymbol{t} := \arg\min_{\tilde{\boldsymbol{t}} \in [k]^{d-1}} \|Q(2j, d, \tilde{\boldsymbol{t}})\|_2$, and let $\boldsymbol{t}'$ denote the prefix that minimizes $\|Q(2j, d, \tilde{\boldsymbol{t}})\|_2$ subject to the constraint that $\boldsymbol{t}'$ must differ from $\boldsymbol{t}$ in the last (i.e. $(d-1)_{th}$) position, i.e.

$$\boldsymbol{t}' = \arg\min_{\tilde{\boldsymbol{t}}' \in [k]^{d-1}, \boldsymbol{t}'_{d-1} \neq \boldsymbol{t}_{d-1}} Q(2j, d, \tilde{\boldsymbol{t}}').$$

The motivation for such choices of $\boldsymbol{t}, \boldsymbol{t}'$ is that since they differ at least by the last position which is an open bracket, they must lead to different next-word distributions. Note also that $P_{d,j}[\bar{\theta}^{(2)}] = \|Q(2j, d, \boldsymbol{t}')\|$.

With the above definition of $\boldsymbol{t}, \boldsymbol{t}'$, consider two valid Dyck prefixes p_1 and p_2 with length no longer than N, defined as follows: for any $d, d' \in [D], i, j \in [k]$, consider a common prefix

$$p = \underbrace{\tau_{2i-1} \dots \tau_{2i-1}}_{d' \text{ open brackets}} \underbrace{\tau_{2i-1}\tau_{2i} \dots \tau_{2i-1}\tau_{2i}}_{(\lfloor \frac{N}{2} \rfloor - d' - d - 1) \text{ pairs}} \underbrace{\tau_{2i} \dots \tau_{2i}}_{d' \text{ closed brackets}}$$

where τ_i denotes a token with type i whose depth is implicit from the context. Set p_1, p_2 as

$$\begin{aligned} p_1 &= p \oplus \boldsymbol{t} \oplus \tau_{2j-1}\tau_{2j}, \\ p_2 &= p \oplus \boldsymbol{t}' \oplus \tau_{2j-1}\tau_{2j}. \end{aligned}$$

That is, p_1, p_2 differ in the last unmatched open bracket. In the following, we will show that the approximate balance condition must hold for the predictions on p_1, p_2 to be sufficiently different.

Bounding the difference in Transformer outputs. For a Transformer $\mathcal{T}$ with second layer parameters $\bar{\theta}_N^{(2)}$, with $\mathcal{P}_{\text{next}}(p)$ indicating the next token probability given a prefix p, by triangle inequality, its outputs on p_1, p_2 satisfy

$$\begin{aligned} &\|\mathcal{T}[\bar{\theta}_N^{(2)}](p_1) - \mathcal{T}[\bar{\theta}_N^{(2)}](p_2)\|_2 \\ &\geq \|\mathcal{P}_{\text{next}}(p_1) - \mathcal{P}_{\text{next}}(p_2)\|_2 - \Big(\|\mathcal{T}[\bar{\theta}_N^{(2)}](p_1) - \mathcal{P}_{\text{next}}(p_1)\|_2 + \|\mathcal{T}[\bar{\theta}_N^{(2)}](p_2) - \mathcal{P}_{\text{next}}(p_2)\|_2\Big) \end{aligned} \tag{2.71}$$

Bounding each term separately:

$$\|\mathcal{P}_{\text{next}}(p_1) - \mathcal{P}_{\text{next}}(p_2)\|_2 \geq \frac{1}{\sqrt{2k}} \|\mathcal{P}_{\text{next}}(p_1) - \mathcal{P}_{\text{next}}(p_2)\|_1 = \frac{1}{\sqrt{2k}} \text{TV}(p_1, p_2)$$

where $\text{TV}(p_1, p_2)$ denotes the TV distance in the next-word distributions from p_1 and p_2, and

$$\|\mathcal{T}[\bar{\theta}_N^{(2)}](p_1) - \mathcal{P}_{\text{next}}(p_1)\|_2 \leq \sqrt{\epsilon}$$

because $\mathcal{L}(\mathcal{T}[\bar{\theta}_N^{(2)}], \mathcal{D}_{\mathsf{Dyck}}) \leq \frac{q(1-q)}{k^2})^N \epsilon$ and the probability of sampling any prefix p is greater than $(\frac{q(1-q)}{k^2})^N$, implying that the per sample next-token squared loss on prefix p is no greater than ϵ. Likewise

$$\|\mathcal{T}[\bar{\theta}_N^{(2)}](p_2) - \mathcal{P}_{\text{next}}(p_2)\|_2 \leq \sqrt{\epsilon}$$

Plugging into Equation (2.71),

$$\|\mathcal{T}[\bar{\theta}_N^{(2)}](p_1) - \mathcal{T}[\bar{\theta}_N^{(2)}](p_2)\|_2 \geq \frac{1}{\sqrt{2k}} \text{TV}(p_1, p_2) - 2\sqrt{\epsilon} \tag{2.72}$$

Define by A_p the contribution of p to the attention output (before LayerNorm) of the last position of p_1, p_2:

$$\begin{aligned} A_p = &\sum_{1 \leq d'' < d'} \left(u(\tau_{2j,d-1}, \tau_{2i,d''-1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d''})\right) \\ &+ \lfloor \frac{N - 2d' - 2d}{2} \rfloor \left(u(\tau_{2j,d-1}, \tau_{2i,d'}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+1})\right). \end{aligned} \tag{2.73}$$

The attention outputs (before LayerNorm) of p_1, p_2, denoted by $A(p_1)$ and $A(p_2)$, satisfy that

$$\begin{aligned} \mathcal{P}_\perp A(p_1) &= \mathcal{P}_\perp (A_p + Q(2j, d, \boldsymbol{t})), \\ \mathcal{P}_\perp A(p_2) &= \mathcal{P}_\perp (A_p + Q(2j, d, \boldsymbol{t}')). \end{aligned} \tag{2.74}$$

Note that for any prefix p',

$$\begin{aligned}\mathcal{T}[\bar{\theta}_N^{(2)}](p') &= \mathrm{g}^{(2)}\big(\mathrm{LN}_C(\mathcal{P}_\perp A(p'))\big) + \boldsymbol{e}(\tau_{2i,d'}) && (2.75)\\ &= \mathrm{g}^{(2)}\Big(\frac{\mathcal{P}_\perp A(p')}{\|\mathcal{P}_\perp A(p')\|}\Big) + \boldsymbol{e}(\tau_{2i,d'}), && (2.76)\end{aligned}$$

where $\mathrm{g}^{(2)}$ is γ-Lipschitz. Hence by Equation (2.76) and Equation (2.72), we have

$$\Big\|\frac{\mathcal{P}_\perp A(p_1)}{\|\mathcal{P}_\perp A(p_1)\|_2} - \frac{\mathcal{P}_\perp A(p_2)}{\|\mathcal{P}_\perp A(p_2)\|_2}\Big\|_2 \geq \frac{\mathrm{TV}(p_1,p_2)}{\sqrt{2k}\gamma} - \frac{2\sqrt{\epsilon}}{\gamma} = \Omega_{\frac{1}{\gamma},\sqrt{\epsilon}}(1). \tag{2.77}$$

Here the TV distance is lower bounded by a constant due to the construction of p_1, p_2, where $\boldsymbol{t}, \boldsymbol{t'}$ differ at the last open bracket.

We will then show that A_p should not be too much larger in norm than $Q(2j, d, \boldsymbol{t})$ or $Q(2j, d, \boldsymbol{t'})$. First, let's state a helper lemma about the contrapositive:

Lemma 2.2.1. *For any $\epsilon > 0$, there exists a constant R_ϵ, such that for any $a, b \in \mathbb{R}^d$ and any $r \in \mathbb{R}^d$ such that $\|r\|_2 \geq R_\epsilon \cdot \max\{\|a\|_2, \|b\|_2\}$, it holds that*

$$\Big\|\frac{a+r}{\|a+r\|_2} - \frac{b+r}{\|b+r\|_2}\Big\|_2 \leq \epsilon.$$

Proof. Denote $r_0 := \max\{\|a\|_2, \|b\|_2\}$. Then $R_\epsilon := \frac{4r_0}{\epsilon} + 1$ suffices:

$$\begin{aligned}&\Big\|\frac{r+a}{\|r+a\|_2} - \frac{r+b}{\|r+b\|_2}\Big\| \leq \|r\| \cdot \Big|\frac{1}{\|r+a\|} - \frac{1}{\|r+b\|}\Big| + \frac{\|a\|}{\|r+a\|} + \frac{\|b\|}{\|r+b\|}\\ \leq&\|r\| \cdot \Big(\frac{1}{\|r\| - r_0} - \frac{1}{\|r\| + r_0}\Big) + \frac{2r_0}{\|r\| - r_0}\\ =&\frac{2r_0}{\|r\| - r_0} \cdot \Big(\frac{\|r\|}{\|r\| + r_0} + 1\Big) \leq \frac{4r_0}{\|r\| - r_0} \leq \frac{4r_0}{R_\epsilon - r_0} \leq \epsilon.\end{aligned}$$

□

Consider Equation (2.77), Equation (2.74), and Lemma 2.2.1 in which

$$\begin{aligned}a &= \mathcal{P}_\perp Q(2j, d, \boldsymbol{t})\\ b &= \mathcal{P}_\perp Q(2j, d, \boldsymbol{t'})\\ r &= \mathcal{P}_\perp A_p\end{aligned}$$

By Lemma 2.2.1, there exists $R_\epsilon \in \mathbb{R}$ such that

$$\|\mathcal{P}_\perp A_p\|_2 \leq R_\epsilon \cdot \max\{\|\mathcal{P}_\perp Q(2j, d, \boldsymbol{t})\|_2, \|\mathcal{P}_\perp Q(2j, d, \boldsymbol{t'})\|_2\}$$

in order for Equation (2.77) to hold. Note that by definition in Equation (2.53),

$$\|Q(2j, d, \boldsymbol{t})\|_2 \leq \|Q(2j, d, \boldsymbol{t'})\|_2 \ = P_{d,j}[\bar{\theta}_N^{(2)}]$$

Hence

$$\begin{aligned}\|\mathcal{P}_\perp A_p\|_2 &\leq R_\epsilon \cdot \|\mathcal{P}_\perp Q(2j, d, \boldsymbol{t'})\|_2\\ &\leq R_\epsilon \cdot \|\mathcal{P}_\perp\|_2 \cdot \|Q(2j, d, \boldsymbol{t'})\|_2\\ &= R_\epsilon \cdot P_{d,j}[\bar{\theta}_N^{(2)}] && (2.78)\end{aligned}$$

As Equation (2.78) holds for p with any d, d', if one choose $d' = 1$, this shows

$$\|u(\tau_{2j,d-1}, \tau_{2i,1}) + u(\tau_{2j,d-1}, \tau_{2i-1,2})\|_2 \leq \frac{4R_\epsilon P_{d,j}[\bar{\theta}_N^{(2)}]}{N}. \tag{2.79}$$

Further, it holds that for any $1 < d' \leq d - 1$,

$$\begin{aligned}
&\| \sum_{1 \leq d'' < d'} (u(\tau_{2j,d-1}, \tau_{2i,d''-1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d''})) \\
&+ \lfloor \frac{N - 2d' - 2d}{2} \rfloor (u(\tau_{2j,d-1}, \tau_{2i,d'}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+1})) \|_2 \\
&\leq R_\epsilon P_{d,j}[\bar{\theta}_N^{(2)}] \text{ , and} \\
&\| \sum_{1 \leq d'' < d'+1} (u(\tau_{2j,d-1}, \tau_{2i,d''-1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d''})) \\
&+ \lfloor \frac{N - 2d' - 2d - 2}{2} \rfloor (u(\tau_{2j,d-1}, \tau_{2i,d'+1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+2})) \|_2 \\
&\leq R_\epsilon P_{d,j}[\bar{\theta}_N^{(2)}].
\end{aligned}$$

Then by triangle inequality,

$$\begin{aligned}
&\lfloor \frac{N - 2d' - 2d - 2}{2} \rfloor \| (u(\tau_{2j,d-1}, \tau_{2i,d'+1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+2})) \\
&\qquad - (u(\tau_{2j,d-1}, \tau_{2i,d'}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+1})) \|_2 \leq 2R_\epsilon P_{d,j}[\bar{\theta}_N^{(2)}].
\end{aligned}$$

Because $N \geq 8D$, we have that $\lfloor \frac{N-2d'-2d-2}{2} \rfloor \geq \frac{N}{8}$, hence it holds that

$$\begin{aligned}
&\| (u(\tau_{2j,d-1}, \tau_{2i,d'+1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+2})) \\
&- (u(\tau_{2j,d-1}, \tau_{2i,d'}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+1})) \|_2 \leq \frac{16R_\epsilon P_{d,j}[\bar{\theta}_N^{(2)}]}{N}.
\end{aligned}$$

Combined with Equation (2.79), one can conclude that,

$$S_{d,d',i,j} = \|u(\tau_{2j,d-1}, \tau_{2i,d'-1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'-1})\| \leq \frac{16DR_\epsilon}{N} P_{d,j}[\bar{\theta}_N^{(2)}]. \tag{2.80}$$

The proof is then completed. □

Proof of Corollary 2.2.2. This proof is in fact a direct combination of Theorems 2.2.1 and 2.2.2. By Theorem 2.2.1 we know there exists a weight $\theta^{(2)*}$ that can reach zero loss for arbitrarily length N. Then it holds that $\|\theta_{\lambda,N}\|_2 \leq \|\theta^{(2)*}\|$ as $\theta_{\lambda,N}$ minimizes the regularized loss. Noticing that bounded weight implies bounded Lipschitzness of $\mathrm{g}^{(2)}$, the rest follows as Theorem 2.2.2. □

2.2.5.5 Proof of Theorem 2.2.3

We now show the limitation of interpretability from a single component, using a Lottery-Ticket-style argument by pruning from large random Transformers.

Theorem 2.2.3 (Indistinguishability From a Single Component)**.** *Consider any L-layer Transformer $\mathcal{T}$ (Equation (2.46)) with embedding dimension m, attention dimension m_a, and projection function $\mathrm{g}^{(l)}$ as*

2-layer ReLU MLP with width w, for $l \in [L]$. [29] *For any $\delta \in (0,1)$ and $N \in \mathbb{N}^+$, consider a $4L$-layer* random *Transformer $\mathcal{T}_{large}$ with embedding dimension $m_{\text{large}} = O(m\log(Lm/\delta))$, attention dimension $m_{\text{large},a} = O(m_a L \log\frac{m_a mLN}{\epsilon\delta})$, and projection function $\mathrm{g}_{\text{large}}$ as 4-layer ReLU MLP with width $w_{\text{large}} = O(\max\{m,w\}L\log\frac{wmLN}{\epsilon\delta})$.*

Assume that $\|W\|_2 \le 1$ for every weight matrix W in $\mathcal{T}$, and suppose the weights are randomly sampled as $W_{i,j} \sim U(-1,1)$ for every $W \in \mathcal{T}_{large}$. Then, with probability $1-\delta$ over the randomness of $\mathcal{T}_{large}$, there exists a nonstructural pruning (Definition 2.2.4) of $\mathcal{T}_{large}$, denoted as $\tilde{\mathcal{T}}_{large}$, which ϵ-approximates $\mathcal{T}$ with respect to $\|\cdot\|_{1,2}$ for any input $X \in \mathbb{R}^{m\times N}$ satisfying $\|\boldsymbol{X}\|_{1,2} \le 1$. [30]

Proof. We will first introduce some notation. For vector $x \in \mathbb{R}^a$ and $y \in \mathbb{R}^b$, we will use $x \oplus y$ to denote their concatenation. We will use 0^a to denote the all-zero vector with dimension a. We will also assume without loss of generality that $w \ge 2m$. [31]

We will use $\bar{\boldsymbol{X}}$ to denote $\begin{bmatrix}\boldsymbol{X}\\ 0^{(m_{\text{large}}-m')\times N}\end{bmatrix}$ for $\boldsymbol{X} \in \mathbb{R}^{m'\times N}$ with $m' \le m_{\text{large}}$.

In the following, a *random network* refers to a network whose weights have entries sampled from a uniform distribution, i.e. $W_{i,j} \sim U(-1,1)$ for every weight W in the random network.

We will first recall Lemma 2.2.2 from Pensia et al. (2020) which shows that a pruned 2-layer random network can approximate a linear function.

Lemma 2.2.2 (Approximating a linear function; Theorem 1 of Pensia et al. (2020) restated)**.** *Let $W \in \mathbb{R}^{m'\times m}, \|W\|_2 = O(1)$, then for $\sigma \in \{\mathrm{ReLU}, \mathcal{I}\}$, where $\mathcal{I}$ represents the identity operator, for a random network $g(x) = W_2\sigma(W_1x)$ with $W_2 \in \mathbb{R}^{m'\times h}, W_1 \in \mathbb{R}^{h\times m}$ for hidden dimension $h = O(m\log(\frac{mm'}{\min\{\epsilon,\delta\}}))$, with probability $1-\delta$, there exists boolean masking matrices M_1, M_2, such that for any $x \in \mathbb{R}^w$,*

$$\|(M_2 \odot W_2)\sigma\big((M_1 \odot W_1)x\big) - Wx\| \le \epsilon\|x\|_2,$$

where $\odot$ denotes the Hadamard product.

We then derive two approximation results Lemmas 2.2.3 and 2.2.4 based on Lemma 2.2.2.

Lemma 2.2.3. *Under the setting of Theorem 2.2.3, with probability $1-2\delta/3$, for any $l \in [L], l' \in [4L-1]$, let $\mathcal{T}^{(l)}$ be the l-th layer of $\mathcal{T}$, there exists a pruning of the $(l'-1)$-th and the (l')-th layer $\mathcal{T}_{large}^{(l'-1)}, \mathcal{T}_{large}^{l'}$, named $\tilde{\mathcal{T}}_{large}^{(l'-1)}, \tilde{\mathcal{T}}_{large}^{l'}$ such that when defined on domain $\|\boldsymbol{X}\|_{1,2} \le 2L, \boldsymbol{X} \in \mathbb{R}^{m\times N}$,*

1. *$\tilde{\mathcal{T}}_{large}^{(l'-1)}$ is independent of the last $m_{\text{large}} - m$ rows of the input.*
2. *$\tilde{\mathcal{T}}_{large}^{l'} \circ \tilde{\mathcal{T}}_{large}^{(l'-1)}\left(\bar{\boldsymbol{X}}\right)$ is an $\left(\frac{C}{1000L^2}\right)^{4L-3}\epsilon$-approximation of $\overline{\mathcal{T}^{(l)}(\boldsymbol{X})}$ with respect to $1,2$-norm.*

Lemma 2.2.4. *Under the setting of Theorem 2.2.3, for any matrix $W \in \mathbb{R}^{4m\times 4m}, \|W\|_2 \le 1$, with probability $1-\delta/4$, for any $l' \in [4L]$, there exists a pruning of the l-th layer $\mathcal{T}_{large}^{(l')}$, named $\tilde{\mathcal{T}}_{large}^{(l')}$, such that when defined on domain $\boldsymbol{X} \in \mathbb{R}^{m\times N}$,*

1. *$\tilde{\mathcal{T}}_{large}^{(l')}$ is independent of the last $m_{\text{large}} - 4m$ rows of the input.*
2. *$a(x) = \tilde{\mathcal{T}}_{large}^{(l')}\left(\bar{\boldsymbol{X}}\right)$ is an $\left(\frac{C}{1000L^2}\right)^{4L}\epsilon$-approximation of $\hat{g}(\boldsymbol{X}) = \overline{W\boldsymbol{X}}$ with respect to $1,2$-norm.*

[29] For notational convenience, we assume all layers share the same dimensions and projection functions. The proof can be trivially extended to cases where the dimensions and projection functions are different.

[30] Here the input and output dimension of $\tilde{\mathcal{T}}_{\text{large}}$ is actually m_{large} which is larger than m; additional dimensions are padded with zeroes. The norm constraint can be easily extended to an arbitrary constant.

[31] We can always pad dimensions if w is too small.

The proof of Lemmas 2.2.3 and 2.2.4 is deferred to Section 2.2.5.6 We can now prove the theorem.

We will first show with induction that if we 1) prune the $(2l-1)$-th and $2l$-th layers of $\mathcal{T}_{\text{large}}$ to approximate $\mathcal{T}^{(l)}$ for each $l \in [L]$, and 2) prune the $2L+1$ to $4L$-th layers of $\mathcal{T}_{\text{large}}$ to approximate identity, then the pruned large transformer will be an ϵ-approximation of $\mathcal{T}$ for any input $\|\boldsymbol{X}\|_{1,2} \leq 1$.

We will perform induction on l: Let $\mathcal{T}^{(1:l)}$ define the composition of layer 1 to l, i.e. $\mathcal{T}^{(1:l)}(\boldsymbol{X}) := \mathcal{T}^{(l)} \circ \mathcal{T}^{(l-1)} \circ \cdots \circ \mathcal{T}^{(1)}(\boldsymbol{X})$, and define $\epsilon_l := \left(\frac{C}{1000L^2}\right)^{4L-3-l} \epsilon$. Suppose that $\mathcal{T}^{(1:2l)}_{\text{large}}$ is an ϵ_l-approximation of $\mathcal{T}^{(1:l)}$. Note that $\|\mathcal{T}^{(1:l)}(\boldsymbol{X})\|_{1,2} \leq (l+1)$, since each attention output has a bounded norm of 1 and every weight matrix in projection function g has spectral norm smaller than 1, hence the norm will at most increment 1 (due to residual connection) after each layer. We have that

$$\left\|\tilde{\mathcal{T}}^{(1:2l)}_{\text{large}}\left(\bar{\boldsymbol{X}}\right)\right\|_{1,2} \leq 4l \leq 4L.$$

Then according to Lemma 2.2.13, $\mathcal{T}^{(l+1)}$ is $(1+200L^2/C)$-Lipschitz on the set of intermediate outputs $\{\left(\tilde{\mathcal{T}}^{(1:2l)}_{\text{large}}(\bar{\boldsymbol{X}})\right)_{1:m} \mid \|\boldsymbol{X}\|_{1,2} \leq 1\}$. We also have that $\mathcal{T}^{(1:l)}(\boldsymbol{X})$ is $(1+200L^2/C)^l$-Lipschitz. Now we can apply Lemma 2.2.5 to show that $\mathcal{T}^{(1:2l+2)}_{\text{large}}$ can ϵ'-approximate $\mathcal{T}^{(1:l+1)}$ with

$$\begin{aligned}\epsilon' &= \epsilon_l(1+200L^2/C) + \epsilon\left(\frac{C}{1000L^2}\right)^{4L-3}(1+200L^2/C)^l + \epsilon_l\left(\frac{C}{1000L^2}\right)^{4L-3}\epsilon \\ &\leq \left(\frac{C}{1000L^2}\right)^{4L-4-l}\epsilon = \epsilon_{l+1}.\end{aligned}$$

The induction is then completed and we have the composition of $\tilde{\mathcal{T}}^{i}_{\text{large}}$ for $i \in [2L]$ ϵ_L-approximates the composition of $\mathcal{T}$ with $\epsilon_L = \left(\frac{C}{1000L^2}\right)^{3L-3}\epsilon$. We will then perform another induction showing that the composition of $\tilde{\mathcal{T}}^{i}_{\text{large}}$ for $i \in [2L+l]$ ϵ_{l+L}-approximates $\mathcal{T}$ with $\epsilon_{l+L} = \left(\frac{C}{1000L^2}\right)^{3L-3-l}\epsilon$. Suppose the statement holds for $L-1 \geq l \geq 0$.

The induction step is similar, because we have $\mathcal{T}$ is $(1+200L^2/C)^L$ Lipschitz, by Lemma 2.2.5, it holds that the composition of $\mathcal{T}^{i}_{\text{large}}$ for $i \in [2L+l+1]$ ϵ'-approximates $\mathcal{T}$ with,

$$\begin{aligned}\epsilon' &= \epsilon_{l+L} + \epsilon\left(\frac{C}{1000L^2}\right)^{4L}(1+200L^2/C)^L + \epsilon_{l+L}\epsilon\left(\frac{C}{1000L^2}\right)^{4L} \\ &\leq \epsilon\left(\frac{C}{1000L^2}\right)^{3L-4-l}\epsilon = \epsilon_{L+l+1}.\end{aligned}$$

This concludes the induction and prove the first claim of the theorem. For the second claim, notice that through similar induction steps, we can prune arbitrary layer of $\mathcal{T}_{\text{large}}$ to approximate identity function and obtain the same approximation rate, this concludes the proof for the second claim. □

2.2.5.6 Helper lemmas for Theorem 2.2.3

Error Analysis Our first lemma shows that the composition of ϵ-approximation can approximate the composition of the original function.

Lemma 2.2.4. *Under the setting of Theorem 2.2.3, for any matrix $W \in \mathbb{R}^{4m \times 4m}$, $\|W\|_2 \leq 1$, with probability $1-\delta/4$, for any $l' \in [4L]$, there exists a pruning of the l-th layer $\mathcal{T}^{(l')}_{large}$, named $\tilde{\mathcal{T}}^{(l')}_{large}$, such that when defined on domain $\boldsymbol{X} \in \mathbb{R}^{m \times N}$,*

1. *$\tilde{\mathcal{T}}^{(l')}_{large}$ is independent of the last $m_{\text{large}} - 4m$ rows of the input.*
2. *$a(x) = \tilde{\mathcal{T}}^{(l')}_{large}\left(\bar{\boldsymbol{X}}\right)$ is an $\left(\frac{C}{1000L^2}\right)^{4L}\epsilon$-approximation of $\hat{g}(\boldsymbol{X}) = \overline{W\boldsymbol{X}}$ with respect to $1,2$-norm.*

Proof. One can prune the value matrix on layer l' to zero and the rest is a direct consequence of Lemmas 2.2.2 and 2.2.20. □

Lemma 2.2.5. *Given three metric spaces A, B, C equipped with same metric $\|\cdot\|$. Suppose $f_1 : A \to B, f_2 : B \to C$ are ϵ_1, ϵ_2-approximations of g_1, g_2 with respect to $\|\cdot\|$, where g_1 is a Lipschitz function with constant λ_1 with respect to $\|\cdot\|$ and $\|g_2(x)\| \le \lambda_2 x$, then it holds that, $f_1 \circ f_2$ is an ϵ'-approximation of $g_1 \circ g_2$, with $\epsilon' = (\lambda_2 + \epsilon_1)(\lambda_1 + \epsilon_2) - \lambda_1\lambda_2$*

Proof. For any $x \in \mathbb{R}^{d_1}$, it holds that,

$$\|f_1(x) - g_1(x)\| \le \epsilon_1\|x\|.$$

This then suggests that,

$$\begin{aligned}
&\|f_2(f_1(x)) - g_2(g_1(x))\| \\
\le&\|f_2(f_1(x)) - g_2(f_1(x))\| + \|g_2(f_1(x)) - g_2(g_1(x))\| \\
\le&\epsilon_2\|f_1(x)\| + \lambda_2\|f_1(x) - g_1(x)\| \\
\le&\epsilon_2\|g_1(x)\| + (\lambda_2 + \epsilon_2)\|f_1(x) - g_1(x)\| \\
\le&\left(\epsilon_2\lambda_1 + \epsilon_1\lambda_2 + \epsilon_1\epsilon_2\right)\|x\|
\end{aligned}$$

□

Approximating ReLU MLP We will first show an extension of Lemma 2.2.2, illustrating that a pruned wide 4-layer ReLU MLP can approximate any 2-layer ReLU MLP.

Lemma 2.2.6. *Consider any 2-layer ReLU MLP $g : \mathbb{R}^{4m} \to \mathbb{R}^{4m}$ parameterized by $W_1 \in \mathbb{R}^{4m\times w}, W_2 \in \mathbb{R}^{w\times 4m}, \|W_1\|_2, \|W_2\|_2 \le 2\sqrt{2}$, for any $\delta, \epsilon \in (0,1)$, consider a random 4-layer ReLU MLP f with input and output dimension $4m$ and width $w' = O(w\log(\frac{wm}{\min\{\epsilon,\delta\}}))$ parameterized by $W_{\text{large},i}$, with probability $1-\delta$ over the randomness of weight of f, there exists a nonstructural pruning of f named $\tilde{f}$, such that $\tilde{f}$ is an $\epsilon-$approximation of f with respect to $2-$norm.*

Proof. Choose $\epsilon_0 = \epsilon/8$. We only need to show there exists boolean matrices M_1, M_2, M_3, M_4, such that,

$$\begin{aligned}
&\Big\|\Big(M_4 \odot W_{\text{large},4}\text{ReLU}\big((M_3 \odot W_{\text{large},3})\text{ReLU}\big((M_2 \odot W_{\text{large},2})\text{ReLU}\big((M_1 \odot W_{\text{large},1})x\big)\big)\big)\Big) \\
&\quad - W_2\text{ReLU}\left(W_1\boldsymbol{X}\right)\Big\|_2 \le \epsilon.
\end{aligned}$$

By Lemma 2.2.2, there exists boolean matrices $M_1 \in \mathbb{R}^{w'\times 4m}$ and $M_2' \in \mathbb{R}^{w\times w'}$, such that for any $x \in \mathbb{R}^{4m}$,

$$\|\left(\begin{bmatrix} M_2' \\ 0^{(w'-w)\times w'}\end{bmatrix} \odot W_{\text{large},2}\right)\text{ReLU}\big((M_1 \odot W_{\text{large},1})x\big) - \begin{bmatrix} W_1 x \\ 0^{w'-w}\end{bmatrix}\|_2 \le \epsilon_0\|x\|_2.$$

Hence we can choose $M_2 = \begin{bmatrix} M_2' \\ 0^{(w'-w)\times w'}\end{bmatrix}$ and have $f_1(x) = \text{ReLU}\big((M_2 \odot W_{\text{large},2})\text{ReLU}\big((M_1 \odot W_{\text{large},1})x\big)\big)$ is ϵ_0-approximation of $g_1(x) = \begin{bmatrix}\text{ReLU}(W_1 x) \\ 0^{w'-w}\end{bmatrix}$.

Again by Lemma 2.2.2, there exists boolean matrices $M_3' \in \mathbb{R}^{w'\times w}$ and $M_4 \in \mathbb{R}^{4m\times w'}$, such that for any $y \in \mathbb{R}^w$,

$$\|\left(M_4 \odot W_{\text{large},4}\right)\text{ReLU}\left(\begin{bmatrix} M_3', 0^{w'\times(w'-w)}\end{bmatrix}\begin{bmatrix} y \\ 0^{w'-w}\end{bmatrix}\right) \le \epsilon_0\|y\|_2$$

Hence we can choose $M_3 = \left[M_3', 0^{w' \times (w'-w)}\right]$, and have $f_2(x) = \mathrm{ReLU}\big((M_4 \odot W_{\text{large},4})\mathrm{ReLU}\big((M_3 \odot W_{\text{large},3})x\big)\big)$ is ϵ_0-approximation of $g_2(x) = W_2 x$.

It is also easy to check g_1 and g_2 are both $2\sqrt{2}$-lipschitz and $g_1(0) = 0$. By Lemma 2.2.5, we conclude that $\tilde{f} = f_1 \odot f_2$ is ϵ'-approximation of $g = g_1 \odot g_2$, with $\epsilon' = 4\sqrt{2}\epsilon_0 + \epsilon_0^2 \le \epsilon$. □

This lemma then yields the following corollaries.

Corollary 2.2.1. *Under the setting of Theorem 2.2.3, with probability $1 - \delta/4$, for any $l \in [L], l' \in [4L]$, there exists a pruning of the projection function $\mathrm{g}_{\text{large}}^{(l')}$, named $\mathrm{g}_{\text{large}}^{(\tilde{l'})}$, such that*

1. *$\mathrm{g}_{\text{large}}^{(\tilde{l'})}$ is independent of the last $m_{\text{large}} - m$ dimension of the input.*
2. *$a(x) = \mathrm{g}_{\text{large}}^{(\tilde{l'})}\left(\begin{bmatrix} x \\ 0^{m_{\text{large}}-m}\end{bmatrix}\right)$ is an $\left(\frac{C}{1000L^2}\right)^{4L}\epsilon$-approximation of $\hat{g}(x) = \begin{bmatrix} \mathrm{g}^{(l)}(x) \\ 0^{m_{\text{large}}-m}\end{bmatrix}$ with respect to $2-$norm.*

Proof. One can construct such pruning by pruning the last $m_{\text{large}} - m$ rows of the weight of the last layer and the last $m_{\text{large}} - m$ columns of the weight of the first layer of $\mathrm{g}_{\text{large}}^{(l')}$ to zero and then apply Lemma 2.2.6. □

Approximating Attention Patterns We will now show that the attention pattern can be approximated by pruning random Transformer layers.

Lemma 2.2.7. *For any $\delta, \epsilon \in (0,1)$, for any $W \in \mathbb{R}^m, \|W\|_2 \le 1$, for two random matrix $W_1, W_2 \in \mathbb{R}^{m' \times m}$ where $m' = O(m \log(\frac{m}{\min\{\epsilon,\delta\}}))$, suppose $\boldsymbol{X} \in \mathbb{R}^{m \times N}$, then there exists nonstructural pruning of W_1, W_2, named $\tilde{W_1}, \tilde{W_2}$, such that*

$$\|\boldsymbol{X}^\top \tilde{W_1}^\top \tilde{W_2} \boldsymbol{X} - \boldsymbol{X}^\top W \boldsymbol{X}\|_\infty \le \epsilon \|\boldsymbol{X}\|_{1,2}^2$$

Here we adopt $\|\|_\infty$ in vector sense, meaning the entry with largest absolute value.

Proof. Suppose without loss of generality, $\|\boldsymbol{X}\|_{:,i} \le 1$. According to Lemma 2.2.2, there exists nonstructural pruning of W_1, W_2, named $\tilde{W_1}, \tilde{W_2}$, such that for any $x \in \mathbb{R}^m, \|x\|_2 \le 1$,

$$\|\tilde{W_1}^\top \tilde{W_2} x - Wx\|_2 \le \epsilon.$$

This then suggests that,

$$\|y^\top (\tilde{W_1}^\top \tilde{W_2} x - Wx)\|_2 \le \epsilon \|y\|_2 \le \epsilon.$$

This concludes the proof. □

The next lemma shows how error propogates through the softmax operators.

Lemma 2.2.8. *For any dimension d, suppose $x, y \in \mathbb{R}^d$ satisfies $\|x - y\|_\infty \le \epsilon$, then it holds that,*

$$\sum_{i=1}^d \left| \frac{\exp(x_i)}{\sum_{i=1}^n \exp(x_i)} - \frac{\exp(y_i)}{\sum_{i=1}^n \exp(y_i)} \right| \le \exp(2\epsilon) - 1.$$

Proof. One can observe that,

$$\exp(-\epsilon)\exp(x_i) \le \exp(y_i) \le \exp(\epsilon)\exp(x_i)$$

This then suggests,

$$\frac{\exp(x_i)}{\sum_{i=1}^n \exp(x_i)} \exp(-2\epsilon) \le \frac{\exp(y_i)}{\sum_{i=1}^n \exp(y_i)} \le \exp(2\epsilon) \frac{\exp(x_i)}{\sum_{i=1}^n \exp(x_i)}.$$

Hence,

$$\sum_{i=1}^d \left| \frac{\exp(x_i)}{\sum_{i=1}^n \exp(x_i)} - \frac{\exp(y_i)}{\sum_{i=1}^n \exp(y_i)} \right| \le \max\{\exp(2\epsilon) - 1, 1 - \exp(-2\epsilon)\} = \exp(2\epsilon) - 1.$$

This concludes the proof. □

Approximating Attention Module We will need the following lemma showing there exists a pruning of the value matrix in $\mathcal{T}_{\text{large}}$ such that it has eigenvalues with magnitude $\Theta(1)$.

Lemma 2.2.9. *For a matrix $W \in \mathbb{R}^{m_{\text{large}} \times m_{\text{large}}}$, with probability at least $1 - \frac{\delta}{10L}$, there exists a pruning of W, named W', such that all the nonzero entries is contained in a $d \times d$ submatrix of W' that satisfies that (1) all its eigenvalues are within $(\frac{1}{2}, 1)$, (2) the index of row specifying the submatrix and the index of column specifying the submatrix are disjoint.*

Proof. As $w_{\text{large}} = \Omega(m \log(\frac{dL}{\delta}))$, hence we can split $W_{1:\lceil m_{\text{large}}/2 \rceil, \lceil m_{\text{large}}/2 \rceil + 1 : m_{\text{large}}}$ into $(m \times (m$ blocks, each with width at least $O(\log(\frac{(m}{\delta}))$ [32]. Within each block, with probability $1 - \frac{\delta}{10 L m_{\text{large}}}$, there exists at least one entry that has value at least $\frac{1}{2}$. We can then choose d disjoint entries in W that are all at least $\frac{1}{2}$, indexed with $\{(a_i, b_i)\}_{i \in [d]}$ where $a_i < a_j$ and $b_i < b_j$ for $i < j$. We can then prune all other entries to zero. Consider the submatrix defined by entries (a, b) for $a \in \{a_i\}_{i \in m}$ and $b \in \{b_i\}_{i \in m}$. Then, this submatrix will be diagonal and contains eigenvalues within $(\frac{1}{2}, 1)$. Further $\{a_i\}_{i \in m}$ and $\{b_i\}_{i \in m}$ must be disjoint because $a_i \le \lceil m_{\text{large}}/2 \rceil < b_i$. The proof is then completed. □

We will also prove that LayerNorm with nonzero normalization constant is Lipschitz.

Lemma 2.2.10. *For LayerNorm function defined as $\text{LN}(x) = \frac{\mathcal{P}_\perp x}{\max\{\|\mathcal{P}_\perp x\|_2, C\}}, x \in \mathbb{R}^m$, for any $x, y \in \mathbb{R}^m$, it holds that,*

$$\Big\| \text{LN}(x) - \text{LN}(y) \Big\|_2 \le 2\|x - y\|_2 / C.$$

Proof. We will proceed by a case analysis:

1. If $\|\mathcal{P}_\perp x\|_2, \|\mathcal{P}_\perp y\|_2 \le C$, then $\Big\| \text{LN}(x) - \text{LN}(y) \Big\|_2 = \frac{\|\mathcal{P}_\perp x - \mathcal{P}_\perp y\|_2}{C} \le \frac{1}{C} \|x - y\|_2$.

2. If $\|\mathcal{P}_\perp x\|_2, \|\mathcal{P}_\perp y\|_2 > C$, then $\Big\| \text{LN}(x) - \text{LN}(y) \Big\|_2 = \frac{\|\mathcal{P}_\perp x - \mathcal{P}_\perp y\|_2}{\|\mathcal{P}_\perp y\|_2} + \left| 1 - \frac{\|\mathcal{P}_\perp x\|_2}{\|\mathcal{P}_\perp y\|_2} \right| \le \frac{2}{C} \|x - y\|_2$.

3. If $\|\mathcal{P}_\perp x\|_2 < C$ and $\|\mathcal{P}_\perp y\|_2 > C$, then $\Big\| \text{LN}(x) - \text{LN}(y) \Big\|_2 = \frac{\|\mathcal{P}_\perp x - \mathcal{P}_\perp y\|_2}{\|\mathcal{P}_\perp y\|_2} + \left| \frac{\|\mathcal{P}_\perp x\|_2}{C} - \frac{\|\mathcal{P}_\perp x\|_2}{\|\mathcal{P}_\perp y\|_2} \right| \le \frac{2}{C} \|x - y\|_2$.

The cases exhaust all possibilities, thus the proof is completed. □

Finally, we will need a lemma showing how error accumulates when we consider both attention patterns and the value matrices.

[32] $O(\cdot)$ hides absolute constants arising from the change of basis in the logarithm.

Lemma 2.2.11. *For any dimension d and positive number N, for $P, Q \in \mathbb{R}^{d\times d}$ satisfying that $\|P\|_2, \|Q\|_2 \leq 1$, for any $x \in \mathbb{R}^{d\times N}$, if matrix $A \in \mathbb{R}^{N\times N}, B \in \mathbb{R}^{d\times N}$ satisfy that,*

$$\begin{aligned} \|A - \sigma(x^\top Qx)\|_{1,1} &\leq \epsilon_1. \\ \|B - Px\|_{1,2} &\leq \epsilon_2. \\ \forall i, k \in [N] \sum_{j\in[N]} A_{j,i} &= 1, A_{k,i} \geq 0. \end{aligned}$$

Then it holds that,

$$\begin{aligned} &\|BA - Px\sigma(x^\top Qx)\|_{1,2} \leq (\epsilon_1\|PX\|_{1,2} + \epsilon_2). \\ &\|\mathrm{LN}_C(BA) - \mathrm{LN}_C(Px\sigma(x^\top Qx))\|_{1,2} \leq 2(\epsilon_1\|PX\|_{1,2} + \epsilon_2)/C. \end{aligned}$$

Proof. For any $i \in N$, we will have

$$\begin{aligned} &\left\|(BA)_{:,i} - \left(Px\sigma(x^\top Qx)\right)_{:,i}\right\|_2 \\ =&\left\|\sum_{j\in[N]} A_{j,i}B_{:,j} - \left(\sigma(x^\top Qx)\right)_{j,i}(PX)_{:,j}\right\|_2 \\ \leq&\|\sum_{j\in[N]} A_{j,i}(PX)_{:,j} - \left(\sigma(x^\top Qx)\right)_{j,i}(PX)_{:,j}\Big\|_2 + \|\sum_{j\in[N]} A_{j,i}\left(PX - B\right)_{:,j}\|_2 \\ \leq&\|PX\|_{1,2}\sum_{j\in[N]} |A_{j,i} - \left(\sigma(x^\top Qx)\right)_{j,i} + \|PX - B\|_{1,2} \\ \leq&\|PX\|_{1,2}\|A - \sigma(x^\top Qx)\|_{1,1} + \|PX - B\|_{1,2} \leq \epsilon_1\|PX\|_{1,2} + \epsilon_2. \end{aligned}$$

The rest follows from Lemma 2.2.10

□

A LayerNorm of larger dimension can be made to be functionally equivalent to a LayerNorm of a smaller dimension. Precisely:

Lemma 2.2.12. *Given any dimension $d < d'$, it holds that for any $x \in \mathbb{R}^d$,*

$$\mathrm{LN}_C(\begin{bmatrix}\mathcal{P}_\perp x \\ 0^{d'-d}\end{bmatrix}) = \begin{bmatrix}\mathrm{LN}_C(x) \\ 0\end{bmatrix}.$$

Proof. The proof follows directly from definition. □

We will now formally define attention module.

Definition 2.2.7 (Attention Module)**.** *We will define attention module $a(\boldsymbol{X} \mid W_V, W_K, W_Q)$ as*

$$a(\boldsymbol{X}) = \mathrm{LN}_C\left(W_V\boldsymbol{X}\sigma(\boldsymbol{X}^\top W_K^\top W_Q\boldsymbol{X})\right).$$

Lemma 2.2.13. *Attention module is lipschitz with respect to $1,2$-norm for bounded input. Precisely, consider attention module (Definition 2.2.7)parameterized by $\|W_V\|_2, \|W_K\|_2, \|W_Q\|_2 \leq 1$ with input domain $\|\boldsymbol{X}\|_{1,2} \leq 4L$, $a(\boldsymbol{X})$ is $200L^2/C$-lipschitz with respect to $1,2-$norm.*

Proof. We have that

$$a(\boldsymbol{X}) = \mathrm{LN}_C\left(W_V\boldsymbol{X}\sigma(\boldsymbol{X}^\top W_K^\top W_Q\boldsymbol{X})\right).$$

Choose ϵ to be a sufficiently small constant, such that, $\exp(32L\epsilon) - 1 \leq 64L\epsilon$. Consider $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ satisfying that $\|\boldsymbol{X} - \tilde{\boldsymbol{X}}\|_{1,2} \leq \epsilon$ and $\|\boldsymbol{X}\|_{1,2} \leq 4L, \|\tilde{\boldsymbol{X}}\|_{1,2} \leq 4L$, we will have

$$\begin{aligned}
&\left| \left(\boldsymbol{X}^\top W_K^\top W_Q \boldsymbol{X} - (\tilde{\boldsymbol{X}})^\top W_K^\top W_Q (\tilde{\boldsymbol{X}})\right)_{i,j} \right| \\
=&\left| (\boldsymbol{X}_{:.i} - \tilde{\boldsymbol{X}}_{:,i})^\top W_K^\top W_Q \boldsymbol{X}_{:,j} + (\tilde{\boldsymbol{X}}_{:,i})^\top W_K^\top W_Q (\boldsymbol{X}_{:,j} - \tilde{\boldsymbol{X}}_{:,j}) + (\boldsymbol{X}_{:.i} - \tilde{\boldsymbol{X}}_{:,i})^\top W_K^\top W_Q (\boldsymbol{X}_{:,j} - \tilde{\boldsymbol{X}}_{:,j}) \right| \\
\leq& 8L\epsilon + \epsilon^2 \leq 16L\epsilon.
\end{aligned}$$

By Lemma 2.2.8, this implies,

$$\|\sigma(\boldsymbol{X}^\top W_K^\top W_Q \boldsymbol{X}) - \sigma((\tilde{\boldsymbol{X}})^\top W_K^\top W_Q (\tilde{\boldsymbol{X}}))\|_{1,1} \leq \exp(32L\epsilon) - 1 \leq 64L\epsilon.$$

We also have

$$\begin{aligned}
\|W_V \left(\boldsymbol{X} - \tilde{\boldsymbol{X}}\right)\|_{1,2} \leq \epsilon. \\
\|W_V \boldsymbol{X}\|_{1,2} \leq 4L
\end{aligned}$$

Lemma 2.2.11 then implies that

$$\|a(\boldsymbol{X}) - a(\tilde{\boldsymbol{X}})\|_{1,2} \leq 200L^2\epsilon/C.$$

This then concludes the proof. □

We can now prove that a large Transformer Layer and an attention module of the larger Transformer can be pruned to approximate the attention module of a smaller Transformer Layer module.

Lemma 2.2.14. *Under the setting of Theorem 2.2.3, with probability $1 - \delta/2$, for any $l \in [L], l' \in [4L-1]$, let $a^{(l)}$ be the attention module on the l-th layer of $\mathcal{T}$, there exists a pruning of the $(l'-1)$-th layer $\mathcal{T}_{large}^{(l'-1)}$, named $\tilde{\mathcal{T}}_{large}^{(l'-1)}$ and the attention module on l'-th layer $a_{\text{large}}^{l'}$ named $a_{\text{large}}^{\tilde{l'}}$, such that when defined on domain $\|\boldsymbol{X}\|_{1,2} \leq 2L$,*

1. *$\tilde{\mathcal{T}}_{large}^{(l'-1)}$ is independent of the last $m_{\text{large}} - m$ rows of the input.*
2. *$\left(a_{\text{large}}^{\tilde{l'}} \circ \tilde{\mathcal{T}}_{large}^{(l'-1)} \left(\begin{bmatrix} x \\ 0^{(m_{\text{large}}-m)\times N}\end{bmatrix}\right)\right)_{1:m}$ is an $\left(\frac{C}{1000L^2}\right)^{4L-1}\epsilon$-approximation of $a^{(l)}(x)$ with respect to $1,2$-norm.*
3. *$\left(\tilde{\mathcal{T}}_{large}^{(l'-1)} \left(\begin{bmatrix} x \\ 0^{(m_{\text{large}}-m)\times N}\end{bmatrix}\right)\right)_{1:m}$ is an $\left(\frac{C}{1000L^2}\right)^{4L}\epsilon$-approximation of $\boldsymbol{X}$ with respect to $1,2$-norm.*

Proof. We will use the shorthand $\epsilon_0 = \left(\frac{C}{1000L^2}\right)^{4L}\epsilon$ and prune in the following order. It holds that for $\epsilon \leq 1$, $\exp(8L^2\epsilon_0) - 1 \leq 16L^2\epsilon_0$.

1. We will prune $W_V^{\text{large},(l')}$ according to Lemma 2.2.9 and name the pruned matrix $W_V^{\tilde{\text{large}},(l')}$. By Lemma 2.2.9, all the nonzero entries is contained in a $d \times d$ submatrix of W' that satisfies that all its eigenvalues are within $(\frac{1}{2}, 1)$. We will assume WLOG the submatrix is the one specified by row $1 \dots d$ and column $d+1 \dots 2d$ and name the submatrix as W.
2. We will then prune $\mathcal{T}_{\text{large}}^{(l'-1)}$ according to Lemma 2.2.4 to output ϵ_0-approximation of

$$\boldsymbol{X} \in \mathbb{R}^{m \times N} \to \begin{bmatrix} \boldsymbol{X} \\ W^{-1}\mathcal{P}_\perp W_V^{(l)}\boldsymbol{X} \\ 0^{(m_{\text{large}}-2m)\times N}\end{bmatrix}$$

As W is defined as the submatrix pruned by $W_V^{(t+1)}$, it holds that

$$W_V^{\widetilde{\text{large},(l')}} \begin{bmatrix} \boldsymbol{X} \\ W^{-1}\mathcal{P}_\perp W_V^{(l)}\boldsymbol{X} \\ 0^{(m_{\text{large}}-m)\times N} \end{bmatrix} = \begin{bmatrix} \mathcal{P}_\perp W_V^{(l)}\boldsymbol{X} \\ 0^{(m_{\text{large}}-m)\times N} \end{bmatrix}$$

3. Finally we will prune $W_K^{\text{large},(l')}, W_Q^{\text{large},(l')}$ according to Lemma 2.2.7 to approximate $(W_K^{(l)})^\top W_Q^{(l)}$ up to ϵ_0 error.

we can now calculate the approximation error. For any $\boldsymbol{X} \in \mathbb{R}^{m\times N}$, $\|\boldsymbol{X}\|_{1,2} \le 2L$, suppose

$$\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{X} + \delta_1 \\ W^{-1}\mathcal{P}_\perp W_V^{(l)}\boldsymbol{X} + \delta_2 \\ 0^{(m_{\text{large}}-2m)\times N} \end{bmatrix}$$

Then by our constrution, it holds that $\forall i \in \{1,2\}, \|\delta_i\|_{1,2} \le \epsilon_0\|\boldsymbol{X}\|_{1,2}$.

We would then have

$$W_V^{\widetilde{\text{large},(l')}}\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X}) = \begin{bmatrix} \mathcal{P}_\perp W_V^{(l)}\boldsymbol{X} + W_V^{\widetilde{\text{large},(l')}}\delta_2 \\ 0^{(m_{\text{large}}-m)\times N} \end{bmatrix} \tag{2.81}$$

By our construction, it holds that $\|W_V^{\widetilde{\text{large},(l')}}\delta_2\|_{1,2} \le 2\|\delta_2\|_{1,2} \le 2\epsilon_0\|\boldsymbol{X}\|_{1,2}$.

Further, by the construction of $W_K^{\widetilde{\text{large},(l')}}, W_Q^{\widetilde{\text{large},(l')}}$, it holds that,

$$\begin{aligned} &\Big\| \left(W_K^{\widetilde{\text{large},(l')}}\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X})\right)^\top \left(W_Q^{\widetilde{\text{large},(l')}}\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X})\right) \\ &- (W_K^{(l)}\boldsymbol{X} + W_K^{(l)}\delta_1)^\top (W_Q^{(l)}\boldsymbol{X} + W_Q^{(l)}\delta_1)\Big\|_\infty \le \epsilon_0 \end{aligned} \tag{2.82}$$

As for any $i,j \in [N]$

$$\begin{aligned} &\left|\left((W_K^{(l)}\boldsymbol{X} + W_K^{(l)}\delta_1)^\top (W_Q^{(l)}\boldsymbol{X} + W_Q^{(l)}\delta_1) - (W_K^{(l)}\boldsymbol{X})^\top W_Q^{(l)}\boldsymbol{X}\right)_{i,j}\right| \\ \le &\left|(W_K^{(l)}\boldsymbol{X}_{:,i})^\top (W_Q^{(l)}\delta_1)_{:,j}\right| + \left|(W_K^{(l)}\delta_1)_{:,i}^\top (W_Q^{(l)}\boldsymbol{X})_{:,j}\right| + \left|(W_K^{(l)}\delta_1)_{:,i}^\top (W_Q^{(l)}\delta_1)_{:,j}\right| \\ \le &\|\boldsymbol{X}\|_{1,2}^2(2\epsilon_0 + \epsilon^2) \le 4\|\boldsymbol{X}\|_{1,2}^2\epsilon_0. \end{aligned}$$

combined with Equation (2.82),

$$\Big\| \left(W_K^{\widetilde{\text{large},(l')}}\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X})\right)^\top \left(W_Q^{\widetilde{\text{large},(l')}}\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X})\right) - (W_K^{(l)}\boldsymbol{X})^\top W_Q^{(l)}\boldsymbol{X}\Big\|_\infty \le \epsilon_0(1 + 4\|\boldsymbol{X}\|_{1,2}^2). \tag{2.83}$$

By Lemma 2.2.8, this implies

$$\begin{aligned} &\Big\|\sigma\left(\left(W_K^{\widetilde{\text{large},(l')}}\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X})\right)^\top \left(W_Q^{\widetilde{\text{large},(l')}}\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X})\right)\right) \\ &- \sigma\left((W_K^{(l)}\boldsymbol{X})^\top W_Q^{(l)}\boldsymbol{X}\right)\Big\|_{1,1} \le 4\epsilon_0(1 + 4\|\boldsymbol{X}\|_{1,2}^2). \end{aligned} \tag{2.84}$$

By Lemma 2.2.11, Equations (2.81) and (2.84) imply,

$$\begin{aligned} &\Big\|W_V^{\widetilde{\text{large},(l')}}\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X})\sigma\left(\left(W_K^{\widetilde{\text{large},(l')}}\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X})\right)^\top \left(W_Q^{\widetilde{\text{large},(l')}}\mathcal{T}^{(\widetilde{l'}-1)}(\boldsymbol{X})\right)\right) \\ &- \begin{bmatrix} \mathcal{P}_\perp W_V^{(l)}\boldsymbol{X}\sigma\left((W_K^{(l)}\boldsymbol{X})^\top W_Q^{(l)}\boldsymbol{X}\right) \\ 0^{(m_{\text{large}}-m)\times N} \end{bmatrix}\Big\|_{1,2} \le 8\epsilon_0(1 + 4\|\boldsymbol{X}\|_{1,2}^2)\|\boldsymbol{X}\|_{1,2} \le 80L^2\epsilon_0. \end{aligned}$$

Now according to Lemmas 2.2.10 and 2.2.12, it holds that

$$\|a^{l'\tilde{}}_{\text{large}} \circ \tilde{\mathcal{T}}^{(l'-1)}_{\text{large}} \left(\begin{bmatrix} x \\ 0^{(m_{\text{large}}-m)\times N} \end{bmatrix} \right)_{1:m} - a^{(l)}(x)\|_{1,2} \leq 160L^2\epsilon_0/C.$$

This concludes the proof. □

Approximating Transformer Layers We will finally show that two random Transformer layers can be pruned to approximate a given Transformer layer.

Lemma 2.2.3. *Under the setting of Theorem 2.2.3, with probability $1 - 2\delta/3$, for any $l \in [L], l' \in [4L-1]$, let $\mathcal{T}^{(l)}$ be the l-th layer of $\mathcal{T}$, there exists a pruning of the $(l'-1)$-th and the (l')-th layer $\mathcal{T}^{(l'-1)}_{large}, \mathcal{T}^{l'}_{large}$, named $\tilde{\mathcal{T}}^{(l'-1)}_{large}, \tilde{\mathcal{T}}^{l'}_{large}$ such that when defined on domain $\|\boldsymbol{X}\|_{1,2} \leq 2L, \boldsymbol{X} \in \mathbb{R}^{m\times N}$,*

1. *$\tilde{\mathcal{T}}^{(l'-1)}_{large}$ is independent of the last $m_{\text{large}} - m$ rows of the input.*
2. *$\tilde{\mathcal{T}}^{l'}_{large} \circ \tilde{\mathcal{T}}^{(l'-1)}_{large}\left(\bar{\boldsymbol{X}}\right)$ is an $\left(\frac{C}{1000L^2}\right)^{4L-3}\epsilon$-approximation of $\overline{\mathcal{T}^{(l)}(\boldsymbol{X})}$ with respect to $1,2$-norm.*

Proof. We will prune the $(l'-1)$-th layer and the attention module of the l'-th layer according to Lemma 2.2.14 to approximate $a^{(l)}$ and the projection function of the l'-th layer according to Corollary 2.2.1. Notice that $\left\|a^{(l)}(\boldsymbol{X}) + \boldsymbol{X}\right\|_{1,2} \leq (\frac{2}{C}+1)\|\boldsymbol{X}\|_{1,2}$ and $\mathrm{g}^{(l)}$ is 1−lipschitz, according to Lemma 2.2.5,

$$\left(\tilde{\mathcal{T}}^{l'}_{\text{large}} \odot \tilde{\mathcal{T}}^{(l'-1)}_{\text{large}} \left(\begin{bmatrix} x \\ 0^{(m_{\text{large}}-m)\times N} \end{bmatrix} \right)\right)_{1:m}$$

is an ϵ'-approximation of $\mathcal{T}^{(l)}(x)$, with

$$\epsilon' \leq (\frac{2}{C}+1)\left(\frac{C}{1000L^2}\right)^{4L}\epsilon + \left(\frac{C}{1000L^2}\right)^{4L-2}\epsilon + \left(\frac{C}{1000L^2}\right)^{8L-2}\epsilon^2 \leq \left(\frac{C}{1000L^2}\right)^{4L-3}\epsilon.$$

This concludes the proof. □

2.2.5.7 Additional lemmas

Lemma 2.2.15. *Given any dimension d and number of samples n, for any size-n dataset $\{(\boldsymbol{x}_i, y_i)\}_{i\in[n]}$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, there exists a width-$2n$ two-layer MLP $f : \mathbb{R}^d \to \mathbb{R}$ with ReLU activation such that, $f(\boldsymbol{x}_i) = y_i$ for any $i \in [n]$.*

Proof. We will first choose direction $w \in \mathbb{R}^d, \|w\|_2 = 1$ and margin $\gamma > 0$ such that for any $i \neq j$ in $[n]$, it holds that,

$$\left|\langle w, \boldsymbol{x}_i - \boldsymbol{x}_j\rangle\right| \geq 2\gamma.$$

We will assume WLOG $w^\top \boldsymbol{x}_i$ is increasing in i.

Then we will construct an auxilliary series z_i for $i \in [n]$ such that,

$$z_1 = y_1/\gamma$$
$$z_i = y_i/\gamma - 2\sum_{j=1}^{i-1} z_j, i \in \{2, \ldots n\}.$$

Finally consider the following two-layer MLP with ReLU activation,

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} z_i \text{ReLU}\left(\langle w, \boldsymbol{x} - \boldsymbol{x}_i \rangle + \gamma\right) - z_i \text{ReLU}\left(\langle w, \boldsymbol{x} - \boldsymbol{x}_i \rangle - \gamma\right),$$

we will show that $f(\boldsymbol{x}_i) = y_i$ for any $i \in [n]$. Notice that

$$z_j \text{ReLU}\left(\langle w, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle + \gamma\right) - z_j \text{ReLU}\left(\langle w, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle - \gamma\right) = \begin{cases} 0, & j > i, \\ \gamma z_i, & j = i, \\ 2\gamma z_j, & j < i. \end{cases}$$

Thus it holds,

$$\begin{aligned} f(\boldsymbol{x}_i) &= \sum_{j=1}^{n} z_j \text{ReLU}\left(\langle w, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle + \gamma\right) - z_j \text{ReLU}\left(\langle w, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle - \gamma\right) \\ &= \sum_{j=1}^{i-1} 2\gamma z_j + \gamma z_i = y_i. \end{aligned}$$

□

Lemma 2.2.16. *Given any sets $\{x_i\}_{i \in m}$ satisfying that $x_i \in \mathbb{R}^n$ and $x_i \neq 0$, there exists a set of orthonormal vectors $\{u_j\}_{j \in [n-2]}$ of $\mathbb{R}^n$ such that (1) $u_j^\top 1^n = 0$ for any $j \in [n-2]$ and (2) $\sum_{j \in [n-2]} u_j^\top x_i u_j \neq x_i$ for any $i \in [m]$.*

Proof. There exists a vector $v \in \mathbb{R}^n$ such that $v^\top x_i \neq 0$ for any $i \in [m]$. We can then construct an orthonormal basis $\{u_j\}_{j \in [n-2]}$ of $\mathbb{R}^n$ as the basis of the normal space of span$(v, 1^n)$. Then the lemma holds. □

Lemma 2.2.17. *Given any dimension n and constant M, there exists a 2-layer width-2n ReLU network $f : \mathbb{R}^{n+1} \to \mathbb{R}$ such that for any $x \in [0, M]^n, y \in [n]$, $f(x \oplus y) = x_y$.*

Proof. The construction is as followed, we will choose f as

$$f(x \oplus y) = \sum_{i=1}^{n} \text{ReLU}(x_i + M(y - i)) - \sum_{i=1}^{n} \text{ReLU}(x_i + M(y - i - 1)) - M(y - 1).$$

Then as we have

$$\text{ReLU}(x_i + M(y - i)) - \sum_{i=1}^{n} \text{ReLU}(x_i + M(y - i - 1)) = \begin{cases} M, & i \leq y - 1; \\ x_i, & i = y; \\ 0, & i \geq y + 1. \end{cases}$$

The proof is completed. □

Lemma 2.2.18. *Given any dimension n and constant $M > 0$, there exists a 2-layer width-2n ReLU network $f : \mathbb{R}^n \to \mathbb{R}$ such that for any $x \in \mathbb{R}^n$ satisfying there exists $i \in [n]$, $x_i > M$ and $\forall j \neq i, x_j = 0$, it holds that $f(x) = i$.*

Proof. The construction is as followed, we will choose f as

$$f(x) = \sum_{i=1}^{m} i \left(\text{ReLU}(x_i) - \text{ReLU}(x_i - M) + M\right)/M.$$

The proof is completed. □

Lemma 2.2.19. *Given any dimension n and natural numbers K, m, M, if there exists K different 2-layer width-m ReLU networks $f_k : \mathbb{R}^n \to \mathbb{R}$, then there exists a 2-layer width-$2Km$ ReLU network $f : \mathbb{R}^{n+1} \to \mathbb{R}$, such that $f(\begin{bmatrix} k \\ x \end{bmatrix}) = f_k(x)$ when $x \in [0, M]^n$.*

Proof. Suppose that

$$f_k(x) = \sum_{i=1}^{m} a_{k,i} \mathrm{ReLU}(w_{k,i}^\top x + b_{k,i}) + b_k.$$

Then we can construct

$$\begin{aligned} f(\begin{bmatrix} y \\ x \end{bmatrix}) = &\sum_{k=1}^{K} \sum_{i=1}^{m} a_{k,i} \mathrm{ReLU}(w_{k,i}^\top x + b_{k,i} + M(y-k)) - a_{k,i} \mathrm{ReLU}(w_{k,i}^\top x + b_{k,i} + M(y-k-1)) \\ &+ b_k - c_{k,i} \mathrm{ReLU}(y + 1 - k), \end{aligned}$$

where $c_{k,i}$ satisfies

$$\forall i, k', \sum_{k=1}^{k'} c_{k,i}(k' + 1 - k) = M \sum_{k=1}^{k'-1} a_{k,i}.$$

The proof is then completed. □

Lemma 2.2.20. *Given any dimension n and $W \in \mathbb{R}^{n \times n}$, $\|W\|_2 \le 2$, there exists a 2-layer width-$2n$ ReLU network $f : \mathbb{R}^n \to \mathbb{R}$ such that for any $x \in \mathbb{R}^n$, it holds that $f(x) = Wx$ and both weight matrices parameterizing f has spectral norm less than $2\sqrt{2}$.*

Proof. The construction is straightforward, one can choose

$$f(x) = [I_n, -I_n]^\top \mathrm{ReLU}\left(\begin{bmatrix} Wx \\ -Wx \end{bmatrix} \right).$$

□

2.2.5.8 Discussion on architecture choices

The reader may notice that Equation (2.42) is not the same as the standard GPT architecture,

$$f_l(X; \theta^{(l)}) = \mathrm{g}^{(l)}\Big(\mathrm{LN}\Big(W_V^{(l)} X \sigma\Big(\mathcal{C} + (W_K^{(l)} X)^\top (W_Q^{(l)} X)\Big) + X\Big)\Big). \tag{2.85}$$

We will shortly discuss the impact of considering Equation (2.85) here.

With similar arguments to Theorem 2.2.2 and the necessity part of Theorem 2.2.1, one can prove that similar balance conditions should also hold for a transformer with a layer specified by Equation (2.85) and a minimal first layer that can nearly perfectly generate bounded Dyck languages.

However, the sufficiency part of Theorem 2.2.1 no longer holds, when the balance condition holds, the last column of the term $W_V^{(2)} X \sigma\Big(\mathcal{C} + (W_K^{(2)} X)^\top (W_Q^{(2)} X)\Big)$ will converge to zero when input length converges to infinity. Hence, if not all $\boldsymbol{e}(\tau_{t,d})$ where $\tau_{t,d}$ is a closed bracket aligns with 1^m, then it is impossible for the model to perfectly generate Dyck for arbitrary length. Although it remains possible to refine a sharper condition for standard GPT architecture to perfectly generate Dyck Language, we find considering Equation (2.42) more elegant in theory. We also verify with experiments that our architecture with standard training can learn bounded Dyck language to more than 97% accuracy. Also, the learned attention patterns are also similarly not interpretable as standard architectures.

2.2.6 Related Work

There has been a flourishing line of work on interpretability in natural language processing. Multiple "probing" tasks have been designed to extract syntactic or semantic information from the learned representations (Raganato & Tiedemann, 2018; Liu et al., 2019; Hewitt & Manning, 2019; Clark et al., 2019). However, the effectiveness of probing often intricately depend on the architecture choices and task design, and sometimes may even result in misleading conclusions (Jain & Wallace, 2019; Serrano & Smith, 2019; Rogers et al., 2020; Brunner et al., 2020; Prasanna et al., 2020; Meister et al., 2021). While these challenges do not completely invalidate existing approaches (Wiegreffe & Pinter, 2019), it does highlight the need for more rigorous understanding of interpretability.

Towards this, we choose to focus on the synthetic setup of Dyck whose solution space is easier to characterize than natural languages, allowing us to identify a set of feasible solutions. While similar representational results have been studied in prior work (Yao et al., 2021; Liu et al., 2023a; Zhao et al., 2023), our work emphasizes that theoretical constructions do not resemble the solutions found in practice. Moreover, the multiplicity of valid constructions suggest that understanding Transformer solutions require analyzing the optimization process, which a number of prior work has made progress on (Lu et al., 2021; Jelassi et al., 2022; Li et al., 2023).

Finally, it is worth noting that the challenges highlighted in our work do not contradict the line of prior work that aim to improve *mechanistic interpretability* into a trained model or the training process (Elhage et al., 2021; Olsson et al., 2022; Nanda et al., 2023; Chughtai et al., 2023; Li et al., 2023), which aim to develop circuit-level understanding of a particular model or the training process.

Interpreting Transformer solutions Prior empirical works show that Transformers trained on natural language data can produce representations that contain rich syntactic and semantic information, by designing a wide range of "probing" tasks (Raganato & Tiedemann, 2018; Liu et al., 2019; Hewitt & Manning, 2019; Clark et al., 2019; Tenney et al., 2019; Hewitt & Liang, 2019; Kovaleva et al., 2019; Lin et al., 2019; Wu et al., 2020; Belinkov, 2022; Liu & Neubig, 2022) (or other approaches using the attention weights or parameters in neurons directly Vig & Belinkov, 2019; Htut et al., 2019; Sun & Marasović, 2021; Eldan & Li, 2023). However, there is no canonical way to probe the model, partially due to the huge design space of probing tasks, and even a slight change in the setup may lead to very different (sometimes even seemingly contradictory) interpretations of the result (Hewitt & Liang, 2019). In this work, we tackle such ambiguity through a different perspective—by developing formal (theoretical) understanding of solutions learned by Transformers. Our results imply that it may be challenging to try to interpret Transformer solutions based on individual parameters (Li et al., 2016; Dar et al., 2022), or based on constructive proofs (unless the Transformer is specially trained to be aligned with a certain algorithm, as in Weiss et al., 2021).

Interpreting attention patterns Prior works (Jain & Wallace, 2019; Serrano & Smith, 2019; Rogers et al., 2020; Grimsley et al., 2020; Brunner et al., 2020; Prasanna et al., 2020; Meister et al., 2021; Bolukbasi et al., 2021; Haab et al., 2023, *inter alia*) present negative results on deriving explanations from attention weights using approaches by Vig & Belinkov (2019); Kobayashi et al. (2020, *inter alia*). However, Wiegreffe & Pinter (2019) argues to the contrary by pointing out flaws in the experimental design and arguments of some of the prior works; they also call for theoretical analysis on the issue. Hence, a takeaway from these prior works is that expositions on explainability based on attention requires clearly defining the notion of explainability adopted (often task-specific). In our work, we restrict our main theoretical analysis to the fully defined data distribution of Dyck language (Definition 2.2.1), and define "interpretable attention pattern" as the stack-like pattern proposed in prior theoretical (Yao et al., 2021) and empirical (Ebrahimi et al., 2020) works. These concrete settings and definitions allow us to mathematically state our results and provide theoretical reasons.

Theoretical understanding of representability Methodologically, our work joins a long line of prior works that characterize the solution of neural networks via the lens of simple synthetic data, from class results on RNN representability (Siegelmann & Sontag, 1992; Gers & Schmidhuber, 2001; Weiss et al., 2018; Suzgun et al., 2019; Merrill, 2019; Hewitt et al., 2020), to the more recent Transformer results on parity (Hahn, 2020), Dyck (Yao et al., 2021), topic model (Li et al., 2023), and formal grammars in general (Bhattamishra et al., 2020a; Li & Risteski, 2021; Zhang et al., 2022a; Liu et al., 2023a; Zhao et al., 2023). Our work complements prior works by showing that although representational results can be obtained via intuitive "constructive proofs" that assign values to the weight matrices, the model does not typically converge to those intuitive solutions in practice. Similar messages are conveyed in Liu et al. (2023a), which presents different types of constructions using different numbers of layers. In contrast, we show that there exist multiple different constructions even when the number of layers is kept the same.

There are also theoretical results on Transformers in terms of Turing completeness (Bhattamishra et al., 2020b; Perez et al., 2021), universal approximatability (Yun et al., 2020), and statistical sample complexity (Wei et al., 2021; Edelman et al., 2022), which are orthogonal to our work.

Transformer optimization Given multiple global optima, understanding Transformer solutions requires analyzing the training dynamics. Recent works theoretically analyze the learning process of Transformers on simple data distributions, e.g. when the attention weights only depend on the position information (Jelassi et al., 2022), or only depend on the content (Li et al., 2023). Our work studies a syntax-motivated setting in which both content and position are critical. We also highlight that Transformer solutions are very sensitive to detailed changes, such as positional encoding, layer norm, sharpness regularization (Foret et al., 2021), or pre-training task (Liu et al., 2022a). On a related topic but towards different goals, a series of prior works aim to improve the training process of Transformers with algorithmic insights (Nguyen & Salazar, 2019; Xiong et al., 2020; Liu et al., 2020; Zhang et al., 2020; Li & Gong, 2021, *inter alia*). An end-to-end theoretical characterization of the training dynamics remains an open problem; recent works that propose useful techniques towards this goal include Gao et al., 2023; Deng et al., 2023.

Mechanistic interpretability It is worth noting that the challenges highlighted in our work do not contradict the line of prior works that aim to improve *mechanistic interpretability* into a trained model or the training process (Cammarata et al., 2020; Elhage et al., 2021; Olsson et al., 2022; Nanda et al., 2023; Chughtai et al., 2023; Li et al., 2023; Wang et al., 2023; Zhong et al., 2023): although we prove that components (e.g. attention scores) of trained Transformers do not generally admit intuitive interpretations based on the data distribution, it is still possible to develop circuit-level understanding about a particular model, or measures that closely track the training process, following these prior works.

Interpretable machine learning In even broader contexts of Interpretable Machine Learning in general, Lipton (2017) outlined common pitfalls of interpretability claims, Chen et al. (2022b) recommended reasonable paths forward, and Bilodeau et al. (2022) proved impossibility results on applying some common classes of simple feature attribution methods on rich model classes.

2.2.7 Conclusion

Why interpreting individual components sometimes leads to misconceptions? Through a case study of the Dyck grammar, we provide theoretical and empirical evidence that even in this simple and well-understood setup, Transformers can implement a rich set of "myopically" non-interpretable solutions. This is reflected both by diverse attention patterns and by the absence of task-specific structures in local components. Our results directly imply similar conclusions for more complex Transformer models; see Section 2.2.5.2 for technical details. Together, this work provides definite proof that myopic interpretability, i.e. methods based on examining individual components only, are not sufficient for understanding the functionality of a trained Transformer.

Our results do not preclude that interpretable attention patterns can emerge; however, they do suggest that interpretable patterns can be infrequent. We discuss the implications for multi-head, overparameterized Transformers trained on more complex data distributions in Section 2.2.4. Moreover, our current results pertain to the existence of solutions; an interesting next step is to study how “inductive biases” given by the synergy of the optimization algorithm and the architecture affect the solutions found.

Chapter 3

Improving language model inference scaling through verifier-guided backtracking

Recently, a plethora of works have proposed inference-time algorithms (e.g. best-of-n), which incorporate verifiers to assist the generation process. Their quality-efficiency trade-offs have been empirically benchmarked on a variety of constrained generation tasks, but the algorithmic design landscape is still largely poorly understood. In this chapter, we develop a mathematical framework for reasoning about constrained generation using a pre-trained language model generator oracle and a process verifier—which can decide whether a prefix can be extended to a string which satisfies the constraints of choice. We show that even in very simple settings, access to a verifier can render an intractable problem (information-theoretically or computationally) to a tractable one. In fact, we show even simple algorithms, like tokenwise rejection sampling, can enjoy significant benefits from access to a verifier. Extending beyond this simple algorithm, we propose a new process-guided test-time sampling algorithm, `VGB`, which uses theoretically grounded *backtracking* to achieve *provably* better robustness to verifier errors. `VGB` interprets autoregressive generation as a random walk on a tree of partial generations, with transition probabilities guided by the process verifier and base model; crucially, backtracking occurs probabilistically. This process generalizes the seminal *Sinclair-Jerrum random walk* (Sinclair & Jerrum, 1989) from the literature on approximate counting and sampling in theoretical computer science, and a conceptual contribution of our work is to highlight parallels with this literature.

Empirically, we demonstrate on both synthetic and real language modeling tasks that a natural modification of tokenwise rejection sampling, in which the sampler is allowed to "backtrack" (i.e., erase the final few generated tokens) has robust and substantive benefits over natural baselines (e.g. (blockwise) rejection sampling, nucleus sampling)—both in terms of computational efficiency, accuracy and diversity. [1] Moreoever, the theoretically provable algorithm, `VGB`, can better mitigate *error amplification* during the course of generation.

In Section 3.1 (based on Botta et al. (2025)), we introduce *query complexity*, our mathematical framework for reasoning about the efficiency of verifier-assisted language generation. In Section 3.2 (based on Botta et al. (2025)), we introduce a simple heuristic algorithm, Tokenwise rejection sampling with backtracking , motivated by our theoretical analysis. In Section 3.3 (based on Rohatgi et al. (2025)), we introduce a stochastic backtracking-based sampling algorithm with theoretical guaranty. Our results suggest that backtracking is a useful component in the algorithmic design space of verifier-assisted language generation, and develop a foundation for theoretically reasoning about these algorithms and improving their theoretical guaranty.

[1] Our codes are released at `https://github.com/YuchenLi01/LM_Query_Complexity`

3.1 Theoretical framework: query complexity of verifier-assisted language generation

The fast-evolving area of inference-time algorithms concerns itself with leveraging the already-impressive capabilities of language models (Raffel et al., 2020; Brown et al., 2020; Touvron et al., 2023), together with a *verifier* which can score generations of the language model. In the simplest form, called *best-of-N*, the language model generates N candidate responses, which are then scored by the verifier, and the highest-scored candidate response is chosen as the output of the inference process (Cobbe et al., 2021; Nakano et al., 2022). If the verifier can score partial generations (sometimes called *process reward*), the space for inference-time algorithms gets much richer: e.g., the final answer can be generated incrementally, using the verifier to guide the process (e.g., by incremental (blockwise) best-of-N, or more complicated strategies like Monte-Carlo-Tree-Search (Browne et al., 2012; Hao et al., 2023)). Importantly, though a flurry of recent papers consider "scaling laws" of natural strategies, the algorithm design space of verifier-aided inference-time algorithms is still opaque. In particular, the *value of a verifier*—and the *relationship it needs to have to the generator* is not well understood.

In this section (based on Botta et al. (2025)), we show that a good verifier can substantially (both in theory and in practice) decrease the computational cost of natural generation tasks, using a pre-trained language model as an *oracle*. In particular, we show that:

- Even simple *constrained generation tasks*—where we are trying to generate a string in the support of a language oracle, subject to some structural constraint (e.g. describable as a simple formal language, like a regular language)—can be *computationally intractable in the absence of a verifier.*

- Conversely, access to a good *process verifier*, one that can decide whether a prefix can be completed to a constraint-satisfying string, can remove these intractabilities. Moreover, even simple algorithms like tokenwise rejection sampling—wherein we generate the string one token at a time, using the process verifier as a means to accept or reject—can have substantive computational benefits over the baseline of rejection sampling.

- Finally, on natural constrained generation tasks—namely, generating test cases for Python functions with a pretrained CodeLlama (Roziere et al., 2023)—a *verifier can be trained*, such that a simple, but natural generalization of tokenwise rejection sampling which is allowed to "backtrack" the last few generated tokens, achieves substantial benefits in computational efficiency, accuracy, and diversity of the generations.

3.1.1 Technical setup

Throughout, we let Σ be a nonempty finite set, denoting the vocabulary. We denote as Σ^i the set of strings of length i and by $\Sigma^* = \cup_{i \in \mathbb{N}} \Sigma^i$ the set of all finite strings on Σ. Given a string $s \in \Sigma^*$, we denote as s_i its i-th element and as $s_{i:j}$ the substring of s starting at its i-element and ending at its j-element, included. We use $|s|$ to denote the length of string s and ϵ to denote the empty string. Finally, we let $x \circ y$ denote the concatenation of string x followed by string y.

Definition 3.1.1 (Autoregressive oracle)**.** *An* ***autoregressive oracle*** $\mathcal{O}$ *takes as input a string* $s \in \Sigma^*$ *and returns a sample from a next-token distribution* $\mathcal{O}(s) : \Sigma \to \mathbb{R}^+$*.*

We will denote the corresponding joint distribution over strings $s \in \Sigma^*$ *as* $p_{\mathcal{O}} : \Sigma^* \to \mathbb{R}^+$*. Correspondingly,* $\forall s \in \Sigma^*$*, let* $p_{\mathcal{O}}(\cdot \mid s)$ *denote the distribution over completions of* s *predicted by* $\mathcal{O}$*.*

Definition 3.1.2 (Constrained generation)**.** ***Constrained generation*** *with respect to an oracle* $\mathcal{O}$*, a constraint set* A*, and vocabulary* Σ *is the task of producing an element* $s \in A \subseteq \Sigma^*$ *such that* $p_{\mathcal{O}}(s) > 0$*. If no such* s *exists, the algorithm needs to output* `FAIL`*.*

When not clear from context, we will specify instances of this task by the triple $(\Sigma, A, \mathcal{O})$. Under suitable choices of the vocabulary Σ and the target domain A, one recovers several language modeling tasks of theoretical and practical relevance as special cases of constrained generation. Specifically, our experiments consider the tasks of generating (i) valid strings under the Dyck grammar (Section 3.2.1) and (ii) valid test cases for a given Python functions (Section 3.2.2), where the oracles return samples from an appropriately pretrained language model. We recover these tasks from Definition 3.1.2 by setting:

- (i) Σ as the set of open and close parentheses, and A as the set of valid sequences of given length.
- (ii) Σ as a set of characters from the Unicode standard (possibly after tokenization) and A as the set of strings that are valid test cases for an input function in the Python programming language.

Note that this task is easier than the task of sampling according to the *restricted distribution* $p(s) \propto \mathbf{1}(s \in A)p_{\mathcal{O}}(s)$, which asks that the relative weights of the strings $s \in A$ that are generated match the probabilities assigned by $p_{\mathcal{O}}$. However, in many settings—e.g., generating a proof of a mathematical problem, or code that performs some intended functionality—we merely care about producing one good sample.

We will be considering "process verifiers" that take as input a prefix s, and output whether or not such a prefix can be completed to a string $s \circ s' \in A$. This is a natural formalization of a "process reward", as it assigns a belief to a partial generation. In the theoretical results (Section 3.1.2 and 3.1.3), we'll assume access to such an idealized verifier. In the empirical results (Section 3.2), such a verifier will be trained and will output a value between 0 and 1, which can be naturally interpreted as a probability that the prefix s is completable to a string $s \circ s' \in A$.

Definition 3.1.3 (Process verifier)**.** *Given a constraint set A, a verifier is a function $V : \Sigma^* \to \{0, 1\}$ such that $\forall s \in \Sigma^*$, $V(s) = 1$ if and only if $\exists s' \in \Sigma^*$ such that $s \circ s' \in A$.*

Designing algorithms given access to oracles which perform certain tasks, is a classical tool in computer science (this is the basis of Turing reductions in computational complexity), as well as optimization (e.g., zero-order optimization assumes a value oracle for a function, first-order optimization a gradient oracle, etc.) In the context of generative modeling, analyses based on oracle complexity have been carried out in the settings of diffusion models, where sampling algorithms rely on score oracles Chen et al. (2022a).

We will consider several natural algorithms that use an autoregressive oracle and a (process) verifier:

Definition 3.1.4 (Rejection sampling)**.** *Rejection sampling works by repeatedly generating a string s according to $p_{\mathcal{O}}$, then running a verifier V on the complete string—and accepting when the verifier outputs $V(s) = 1$.*

Note, this algorithm only needs a verifier that decides the membership in A, rather than a process verifier. On the other hand, because the entire string needs to be generated first before being verified—the number of generations until the verifier accepts is likely very large.

Definition 3.1.5 (Tokenwise rejection sampling)**.** *Tokenwise rejection sampling works by generating a string one token at a time. To generate the next token t, given a prefix s, we sample $t \sim \mathcal{O}(s)$, and run the process verifier on $V(s \circ t)$. We repeat this, until $V(s \circ t) = 1$, then proceed to the next token.*

This algorithm requires a process verifier. However, since a partial string is accepted only if the process verifier accepts, the number of generations needed is likely to be smaller. In fact, we provide a very simple example in Section 3.1.3.

When arguing about lower bounds, a natural lower bound on the complexity of an algorithm is the number of oracle calls needed[2], particularly so when this dominates the cost of the algorithm, as is frequently the case for language models:

Definition 3.1.6 (Oracle complexity)**.** *Given a (possibly randomized) algorithm $\mathcal{A}$ that solves the constrained generation instance $(\Sigma, A, \mathcal{O})$, the **oracle complexity** of $\mathcal{A}$ is defined as the expected number of calls to the oracle made by $\mathcal{A}$ to solve $(\Sigma, A, \mathcal{O})$, namely:*

$$\mathcal{C}(\mathcal{A}) = \mathbb{E}[\#\textit{calls to } \mathcal{O} \textit{ made by running } \mathcal{A}],$$

[2] In our case, the number of calls is a randomized quantity, so a natural quantity to consider is the expected number of oracle calls. It is of course reasonable to consider finer-grained notions like tail bounds on the number of calls.

where the expectation is taken over the randomness of the oracle $\mathcal{O}$ and the randomness of the algorithm $\mathcal{A}$.

Finally, we recall the classical knapsack problem, which will be used in a reduction to prove computational intractability results for the constrained generation task:

Definition 3.1.7 (Knapsack problem)**.** *Given a set of weights $\{X_i \in \mathbb{Z}_{\geq 0} \mid i \in [D]\}$ and $c \in \mathbb{Z}_{\geq 0}$, the knapsack problem seeks an assignment of the variables $(a_i)_{i=1}^D$, with $a_i \in \{0, 1\}$ $\forall i \in [D]$ such that $c = \sum_{i=1}^D a_i X_i$.*

The problem is (weakly) NP-hard, even for some very special choices of c, X_i.

3.1.2 Constrained generation is hard without a verifier

First, we show that the constrained generation task (Definition 3.1.2), without access to a process verifier can be intractable—even if the constraint set A is extremely simple (e.g. the parity of a binary string).

The source of intractability can be *information-theoretic*: namely, if the oracle does not have a succinct description, the algorithm may need to query it prohibitively many times to identify what oracle it's interacting with. We view this as a plausible obstruction in practice as well: language models frequently behave unpredictably "in-the-tails", which becomes increasingly more likely when generating long strings. Thus, to inspect the behavior of the model on long strings, many queries are needed.

The source of the intractability can also be *computational*: namely, even if the oracle is very simple (e.g., a uniform distribution), generating a member of A can be NP-hard, even if checking membership in A can be done efficiently. Perhaps this should not come as a surprise: after all, easy verification of membership, but hard generation, is the hallmark of NP-hard problems.

Proceeding to the first result, we show the following:

Theorem 3.1.1. *There exists a constrained generation task $(\Sigma, A, \mathcal{O})$ for which $\Sigma = \{0, 1\}$, $A \subseteq \Sigma^D$, and $\mathcal{O}$ is an (unknown) member of a set of 2^{D-1} possible oracles, such that any (possibly randomized) algorithm $\mathcal{A}$ has an (expected) oracle complexity of at least 2^{D-1}.*

Intuitively, the lower bound is shown by engineering a scenario such that the behavior of the oracle on long strings is unknown to the algorithm—but success of the generation task relies on "guessing" this behavior correctly.

Proof. Consider the constrained generation task $(\Sigma, A, \mathcal{O}_{\hat{s}})$, such that $\Sigma := \{0, 1\}$, $A := \{s \in \Sigma^D : \sum_{i=1}^D s_i \mod 2 = 0\}$ for some fixed $D \in \mathbb{Z}_+$. Moreover, the oracle $\mathcal{O}_{\hat{s}}$ is indexed by an (unknown to the algorithm) $\hat{s} \in \Sigma^{D-1}$, and it specifies the autoregressive distribution defined s.t. $\forall s \in \Sigma^*, |s| < D - 1$, we have $p_{\mathcal{O}_{\hat{s}}}(1|s) = p_{\mathcal{O}_{\hat{s}}}(0|s) = 1/2$; while for $s \in \Sigma^*, |s| = D - 1$, it satisfies:

$\forall s \neq \hat{s} \in \Sigma^{D-1}, s_D \in \{0, 1\}$, we have:

$$p_{\mathcal{O}_{\hat{s}}}(s_D \mid s) = \begin{cases} 1, \text{ if } \left(\sum_{j=1}^{D-1} s_j + s_D\right) \mod 2 = 1 \\ 0, \text{ otherwise} \end{cases} \tag{3.1}$$

For $s = \hat{s}, s_D \in \{0, 1\}$, we have:

$$p_{\mathcal{O}_{\hat{s}}}(s_D \mid s) = \begin{cases} 1, \text{ if } \left(\sum_{j=1}^{D-1} s_j + s_D\right) \mod 2 = 0 \\ 0, \text{ otherwise} \end{cases} \tag{3.2}$$

Suppose first that the algorithm is deterministic, and we choose the prefix $\hat{s}$ uniformly at random. Let us denote by $x_1, x_2, x_3, \ldots, x_q \in \Sigma^*$ the queries to $\mathcal{O}$ generated by the algorithm. The claim is that expected number of queries q needed to ensure at least one $x_i, i \in [q]$ is in A is 2^{D-1}. Indeed, the x_i s.t. $|x_i| < D - 1$ reveal no information about $\hat{s}$: the output of $\mathcal{O}$ is a uniform Bernoulli random variable regardless of the value of $\hat{s}$. On the other hand, if at some point the algorithm has queried a set S of x_i of length $D - 1$, the probability over $\hat{s}$ is uniform over $\Sigma^{D-1} \setminus S$. Hence, the expected number of queries q (expectation being over the choice of $\hat{s}$) a deterministic algorithm needs is lower bounded by 2^{n-1}.

By Yao's minimax lemma (Yao, 1977), this means that for any (even possibly randomized) algorithm $\mathcal{A}$, there exists $\hat{s}$ on which the algorithm makes at least 2^{n-1} queries in expectation. [3] □

Proceeding to the computational lower bound, the theorem we show is as follows:

Theorem 3.1.2. *There exists a constrained generation task $(\Sigma, A, \mathcal{O})$ for which $\Sigma = \{0, 1\}$, membership in $A \subseteq \Sigma^D$ can be checked in time polynomial in D, and $\mathcal{O}$ is such that $\forall s \in \{0,1\}^D, p_{\mathcal{O}}(s) > 0$, the generation task is NP-hard.*

Proof. We construct a reduction from the knapsack problem (Definition 3.1.7). Let the set $\{X_1, \ldots, X_D\}$ and the integer c specify an arbitrary instance of the knapsack problem. Consider the constrained generation task specified by $\Sigma := \{0, 1\}$, $A := \{s \in \Sigma^D : \forall i \in [D], s_i \in \{0, 1\};\ \sum_{i=1}^{D} s_i X_i = c\}$. Membership in this A can be clearly verified in polynomial time. Suppose we have a poly-time algorithm that generates a solution $\hat{s}$ to $(\Sigma, A, \mathcal{O})$. Since $\forall s \in \Sigma^D, p_{\mathcal{O}}(s) > 0$, $\hat{s}$ provides a solution to the knapsack problem, as we needed. □

3.1.3 Constrained generation with process verifier gets easier

While pessimistic, the message of Section 3.1.2 agrees with recent developments in inference-time scaling: namely, many natural tasks of interest seem to require a verifier to be solved.

First, we show that the simplest "natural" algorithm with a process verifier, tokenwise rejection sampling (Definition 3.1.5), can be much more efficient (exponentially so) in terms of oracle complexity compared to the trivial baseline of rejection sampling (Definition 3.1.4).

Proposition 3.1.1. *Consider the constrained generation task $(\Sigma, A, \mathcal{O})$, s.t. $\Sigma = \{0, 1\}$, $A = \{0^D\}$ and $\mathcal{O}$ is uniform over Σ^D. Then:*

1. *The expected oracle complexity of rejection sampling (Definition 3.1.4) is $2^D D$.*
2. *The expected oracle complexity of tokenwise rejection sampling (Definition 3.1.5) with a perfect process verifier is $2D$.*

Proof. Both claims are straightforward. (1) follows as generating one guess for the string s takes D oracle calls. Moreover, the probability of the full string matching the only string in A (i.e., 0^D) is $1/2^D$. As the number of calls to generate 0^D is a geometric random variable, the expected number of full string generations is 2^D.

For (2), since $\mathcal{O}$ is uniform, at each token, the probability of drawing 0 is $1/2$. Hence, the expected number of calls per coordinate needed is 2 — making the total number of expected calls for the entire string $2D$. □

This proposition underscores the power of a process verifier — even in extremely simple settings, and even when used in conjunction with a very simple algorithm.

In fact, one can easily see that with a perfect process verifier, one can easily solve the constrained generation task with $|\Sigma| D$ calls: at each position, one queries the process verifier for each possible continuation of the string, and accepts only if the process verifier accepts. Of course, in practice, the verifier is not perfect, and its accuracy likely depends on how "out-of-distribution" the prefix it's queried on is (See Section 3.2.1.5 and Section 3.2.2.9)

We finally remark that a process verifier, as we defined it, is clearly useful to solve the generation task. If we instead wanted to sample from the restricted distribution $p(s) \propto \mathbf{1}(s \in A) p_{\mathcal{O}}(s)$, it's not clear how useful the process verifier is. For instance, if we use the simple tokenwise rejection sampling (Definition 3.1.5), it's easy to see that the distribution we produce samples from is *not* the restricted distribution:

[3] We discuss additional intuitions for understanding our proof in Remark 3.5.1 in Section 3.5.2.

Proposition 3.1.2. *Consider the constrained generation task* $(\Sigma, A, \mathcal{O})$, *s.t.* $\Sigma = \{0, 1\}$, $A = \{s \in \Sigma^D : \exists i \in [D], s_i = 0\}$ *and* $\mathcal{O}$ *is uniform over* Σ^D. *Then, tokenwise rejection sampling does* not *produce samples from* $p(s) \propto \mathbf{1}(s \in A)p_{\mathcal{O}}(s)$.

Proof. By Definition 3.1.5, until the last token is being generated, the process verifier will always accept (as there exists a string with at least one 0 coordinate in the coordinates that haven't yet been sampled). Now, for the prefix 1^{D-1}, the only completion that is in A is $1^{D-1} \circ 0$. This means that $1^{D-1} \circ 0$ is assigned probability mass $\frac{1}{2^{D-1}}$ under the tokenwise rejection sampling schema. All other strings in Σ^D are assigned a probability $\frac{1}{2^D}$. On the other hand, $p(s) \propto \mathbf{1}(s \in A)p_{\mathcal{O}}(s)$ assigns uniform mass on all strings in A — proving the claim of the proposition. □

3.2 Heuristic algorithm: tokenwise rejection sampling with backtracking

Our theory (Section 3.1) proves that in some cases, tokenwise rejection sampling algorithm (Definition 3.1.5) with access to a process verifier can be exponentially faster than only verifying at the end (Definition 3.1.4). In this section, we build on the tokenwise rejection sampling algorithm (Definition 3.1.5), and additionally consider a "backtracking" strategy, in which the model is allowed to erase some of its generations. The reasons to consider such a strategy is to allow the model to get "unstuck": if the process verifier decides the current prefix cannot be completed to a valid string in A, it is possible that erasing the last few tokens will make it easier for the model to correct its mistake, compared to erasing just the last token. More formally, the framework of our algorithm is given by Algorithm 1 below. [4]

Algorithm 1 Tokenwise rejection sampling with backtracking

```
1:  Input: Prompt $x$, generator $\mathcal{O}$, verifier $\mathsf{V}$, length $D \in \mathbb{N}_+$, backtrack quota $Q \in \mathbb{N}$, backtrack stride $B \in \mathbb{N}_+$
2:  $s \leftarrow \epsilon$
3:  while $|s| < D$ and $s_{|s|} \neq$ <eos> do
4:      Sample $\hat{s} \sim \mathcal{O}(x \circ s)$
5:      $s \leftarrow s \circ \hat{s}$
6:      if $Q > 0$ and $\mathsf{V}(x \circ s) = 0$ then
7:          $s \leftarrow s_{1:|s|-B}$
8:          $Q \leftarrow Q - 1$
9:          for i in $1 \cdots B$ do
10:             Choose $\hat{s} \in \arg\max \mathcal{O}(x \circ s)$
11:             $s \leftarrow s \circ \hat{s}$
12:         end for
13:     end if
14: end while
```

The flexibility of the tokenwise rejection sampling with backtracking (Algorithm 1) makes it a very natural strategy to use in conjunction with trained verifiers. We perform a thorough empirical investigation into the applicability of Tokenwise rejection sampling with backtracking in constrained language generation, and benchmark it against common baselines, including rejection sampling (Definition 3.1.4), nucleus sampling

[4] The algorithm is a bit more involved, so we will describe it in pseudocode rather than text. Besides the notations in Section 3.1.1, Algorithm 1 uses the following additional common conventions: `<eos>` denotes the end-of-sequence token; $s_{|s|} \neq$ `<eos>` is understood as True when $s = \varepsilon$; for any starting index i and ending index j, if $i > j$, then $s_{i:j} = \varepsilon$. In line 10, why redoing the erased positions using argmax: our results in Section 3.2.1.1 suggests that *out-of-distribution prefix* is a cause of generator mistakes. As a remedy, redoing the erased positions using argmax is intended to increase the generator-predicted probability of the currently sampled prefix. We include an ablation study in Section 3.2.3 verifying that this improves the accuracy.

(Holtzman et al., 2020), temperature scaling, and "block best-of-N" (Section 3.2.2.6) sampling, on both synthetic data (Section 3.2.1) and more realistic data (Section 3.2.2). We observe that across various settings, Tokenwise rejection sampling with backtracking reduces query complexity, improves accuracy, and does not hurt diversity.

3.2.1 Language models trained on synthetic data

3.2.1.1 Dyck grammar as a sandbox

Real-world LLM pretraining data (Li et al., 2024a) typically involves many diverse structures, so when an LLM algorithm outperforms baselines on a benchmark, it is generally challenging to precisely identify which component of the algorithm improved the handling of which structures of the data.

To have a quantitative control over the structure in the pretraining data distribution, and to derive fine-grained observations about the effects of Tokenwise rejection sampling with backtracking , we synthetically generate the pretraining data based on the *Dyck grammar* (Schützenberger, 1963), a classic formal language (context-free grammar) consisting of balanced parentheses of multiple types (for example, "[()]" is valid but "([)]" is not). Dyck serves as a useful sandbox, as it typifies features such as long-range dependencies and a hierarchical, tree-like structure—characteristics often found in both natural and programming language syntax—and has been a subject of interest in numerous theoretical studies on Transformers (Yao et al., 2021; Liu et al., 2023c;a; Wen et al., 2023). More formally:

Definition 3.2.1 (Dyck distribution)**.** Dyck_D *denotes the Dyck language* [5] *of length* D *defined over the vocabulary* $\Sigma = \{ \texttt{[}, \texttt{]}, \texttt{(}, \texttt{)} \}$*, whose* length-$N$ prefix set *is denoted as* $\mathsf{Dyck}_N, \forall N \in [D]$*. For a valid prefix* $w_{1:N} \in \mathsf{Dyck}_N$*, the* depth *of* $w_{1:N}$ *is*

$$\begin{aligned} d(w_{1:N}) = \#&\textit{Open Brackets in } w_{1:N} \\ &- \#\textit{Close Brackets in } w_{1:N}. \end{aligned}$$

The distribution $\mathcal{D}_{\mathsf{Dyck}}$ *over* Dyck_N*, (parameterized by* $p, q \in (0,1)$*) is defined such that* $\forall w_{1:N} \in \mathsf{Dyck}_N$*,*

$$\mathbb{P}(w_{1:N}) \propto p^{|\{i | w_i = \texttt{[}, d(w_{1:i}) = 1\}|} \cdot (1-p)^{|\{i | w_i = \texttt{(}, d(w_{1:i}) = 1\}|} \tag{3.3}$$

$$\cdot (pq)^{|\{i | w_i = \texttt{[}, d(w_{1:i}) > 1\}|} \cdot ((1-p)q)^{|\{i | w_i = \texttt{(}, d(w_{1:i}) > 1\}|} \tag{3.4}$$

$$\cdot (1-q)^{|\{i | w_i \in \{\texttt{]}, \texttt{)}\}, d(w_{1:i}) \le D - i\}|}.$$

Remark 3.2.1. *Equation (3.3) defines an intuitive autoregressive generative process for* Dyck_D*: if the current depth is 0, then sample the next token from* [*and* (*with probability* p *and* $1-p$ *respectively; else if the current depth is* $D - i + 1$*, implying that all the remaining positions have to be closing brackets, then deterministically close the last unmatched open bracket* [6]*; else, sample the next token from open or close brackets with probability* q *and* $1-q$ *respectively. In other words,* p *controls the proportion of square vs. round brackets, while* q *controls the tendency to predict an open bracket when possible (a large* q *may result in a large depth at some position).*

In our experiments, we pretrain autoregressive Transformer (Vaswani et al., 2017) Language models (6 layers, 8 heads per layer, hidden dimension 512) from scratch on data sampled from $\mathcal{D}_{\mathsf{Dyck}}$ with $D = 32, p = 0.2, q = 0.5$. We use batch size 32, weight decay 0.1, learning rate 3e-4 with 100 warmup steps, and follow Block et al. (2024) to use exponential moving average to stabilize training. We reached 100% training and (in-distribution) validation accuracy.

To search for stronger signals in benchmarking the accuracy of the trained model, we will prompt it using the following type of *out-of-distribution* prompts. Note that since $p < 0.5$, the training data contains less square brackets than round brackets, so long prefixes with many square brackets will be *out-of-distribution*

[5] We follow a simplified version of Wen et al. (2023) in defining a probability distribution over strings in a Dyck language.

[6] At any position, there is at most one valid closing bracket.

prompts for the trained model. We generated a set of such out-of-distribution prompts Dyck_{OOD} from Dyck_N with $p = 0.8$ where the prefix length N is uniformly randomly sampled from $25 \leq N \leq 31$. We let the trained language model complete these prompts and check whether the completed string is in Dyck_D. Quantitatively:

Definition 3.2.2 (Prompt completion accuracy)**.** *Given an autoregressive oracle $\mathcal{O}$ (Definition 3.1.1) and a set of prefix prompts X, the accuracy of $\mathcal{O}$ in completing X is:*

$$Acc(\mathcal{O}, X) = \frac{1}{|X|} \sum_{x \in X, y \sim p_{\mathcal{O}}(\cdot|x)} \mathbf{1}_{x \circ y \in \mathsf{Dyck}_D}$$

We construct the autoregressive oracle $\mathcal{O}_{\text{nucleus}}$ which predicts the next-token distribution based on our trained model with nucleus sampling (Holtzman et al., 2020) top_p set to 0.9. We observed that $\text{Acc}(\mathcal{O}_{\text{nucleus}}, \mathsf{Dyck}_{OOD}) = 94.23\%$. We will show that $\mathcal{O}_{\text{verifier backtracking}}$ based on Algorithm 1 can significantly reduce the remaining error rate.

3.2.1.2 Training the verifier

We collect a set of 441 prompts in Dyck_{OOD} in which the trained model (denoted as LM) made mistakes when completing them. We implement a rule-based error parser according to the grammars of Dyck_D which identifies the first position of error in each model completion. Applying this parser to the model mistakes, we obtain a set of model-generated strings $X_{\text{error}} \subset \Sigma^*$ which contain errors. By contrast, we sample another set of 441 strings $X_{\text{correct}} \sim \mathsf{Dyck}_{OOD}$ such that X_{error} and X_{correct} have the same length distribution. We train a lightweight neural network verifier to distinguish X_{error} from X_{correct}.

Concretely, to maximally exploit the representations learned by LM, we train a 1-linear-layer verifier V whose features are the last-layer-last-position representations by LM of strings in $X_{\text{error}} \cup X_{\text{correct}}$, and labels are 0 for strings in X_{error} and 1 for strings in X_{correct}. Consequently, the trainable parameters of V are a single matrix of dimensionality 512 by 2. Among the 882 strings in $X_{\text{error}} \cup X_{\text{correct}}$, we use 792 samples for training, and 90 samples for validation. Despite being slightly over-parameterized, this minimal verifier V achieved on average 93% (with standard error 3.9%) validation accuracy across 10 repetitions. Figure 3.1 in Section 3.2.1.6 illustrates the intuition of why a lightweight verifier may be surprisingly effective with a small number of labeled samples. We next verify that forcing a backtracking at prefixes where the model made mistakes can effectively improve the completion accuracy (Section 3.2.1.3), and that the trained verifier in this section can mostly catch those mistakes and thus mostly retaining the accuracy gain (Section 3.2.1.4).

3.2.1.3 Backtracking effectively reduces errors

The trained language model LM made a mistake at the last position of each string $x \in X_{\text{error}}$. We therefore use "error-inducing prefixes" $X_{\text{error-inducing}}$ to denote $\{x_{1:|x|-1} \mid x \in X_{\text{error}}\}$. Table 3.1 shows that at prefixes in $X_{\text{error-inducing}}$, if we backtrack *only once* for a small backtrack stride B, and continue the autoregressive sampling process, the error rate can be significantly reduced.

3.2.1.4 Verifier effectively reduces errors

In Section 3.2.1.3, the sampling process forced a backtracking at error-inducing prefixes $X_{\text{error-inducing}}$. Can the error reduction effect be retained by a *trained* lightweight single-layer verifier V in Section 3.2.1.2? Table 3.2 shows that Tokenwise rejection sampling with backtracking (Algorithm 1) using the trained verifier is remarkably effective. Moreover, in Section 3.2.1.7, we verify that the predicted backtracks were necessary.

3.2.1.5 Tokenwise rejection sampling with backtracking reduces completion errors on unseen OOD prefixes

Table 3.2 in Section 3.2.1.4 reported a significant improvement of accuracy by Tokenwise rejection sampling with backtracking (Algorithm 1) when the prompts are $X_{\text{error-inducing}}$, for which the language model LM

generation configuration	**accuracy**
baseline: nucleus sampling top_p = 0.9	0.331
baseline: greedy argmax sampling	0.334
$B = 1$, then nucleus sampling top_p = 0.9	0.366
$B = 2$, then nucleus sampling top_p = 0.9	0.438
$B = 4$, then nucleus sampling top_p = 0.9	0.591
$B = 8$, then nucleus sampling top_p = 0.9	0.790

Table 3.1: At error-inducing prefixes, a larger backtrack stride B significantly improves completion accuracy (Definition 3.2.2).

Q	B	**accuracy**
1	2	0.421
	4	0.500
	6	0.604
2	2	0.457
	4	0.634
	6	0.762
4	2	0.518
	4	0.762
	6	0.921
baseline: nucleus sampling top_p = 0.9		0.331
baseline: greedy argmax sampling		0.334

Table 3.2: When the prompts are error-inducing prefixes, a single-layer trained verifier significantly improves completion accuracy using Tokenwise rejection sampling with backtracking (Algorithm 1). A larger backtrack quota Q and a larger backtrack stride B are both helpful.

made mistakes during completion. Is the verifier V overfitted to these type of error-inducing prompts? Can the accuracy improvement generalize to (average-case) out-of-distribution (OOD) prefixes, i.e. independently sampled strings of the same distribution as Dyck_{OOD} (Section 3.2.1.1)?

We independently sampled 10000 such out-of-distribution prompts $\mathsf{Dyck}_{OOD}^{unseen}$, and benchmark the accuracy of Tokenwise rejection sampling with backtracking (Algorithm 1) against the baselines of nucleus sampling top_p = 0.9 (Holtzman et al., 2020) and standard autoregressive sampling (equivalent to top_p = 1.0). Table 3.3 shows that Tokenwise rejection sampling with backtracking (Algorithm 1) significantly reduces completion errors. Crucially, the improvement does not diminish on top of the commonly used baseline of truncating the tail probabilities during sequence sampling. This verifies the desirable property that Tokenwise rejection sampling with backtracking can be applied in combination with such baselines to further improve accuracy. We also verify that the accuracy improvement does not hurt diversity (Section 3.2.1.9).

Finally, provided with the verifier, why does the model still make mistakes? We include additional error analysis in Section 3.2.1.8.

3.2.1.6 Visualizing the language model representations of correct vs. incorrect sequences

nucleus sampling top_p	Q	B	**#errors ± std err**
0.9	0	0	240.0 ± 5.177
	4	4	179.4 ± 1.020
1.0	0	0	461.8 ± 8.304
	4	4	200.0 ± 3.225

Table 3.3: Tokenwise rejection sampling with backtracking (Algorithm 1) reduces completion errors on unseen out-of-distribution (OOD) prefixes. Crucially, the improvement does not diminish on top of commonly used baselines, including nucleus sampling top_p = 0.9. For each setting of top_p, we compare Tokenwise rejection sampling with backtracking (Algorithm 1) (using backtrack quota $Q = 4$ and backtrack stride $B = 4$) with the baseline (using backtrack quota $Q = 0$ and backtrack stride $B = 0$). We report the number of completion errors that occur when completing an unseen set of 10000 independently sampled out-of-distribution prompts $\mathsf{Dyck}_{OOD}^{unseen}$. The experiment was repeated 5 times, and we report the standard errors.

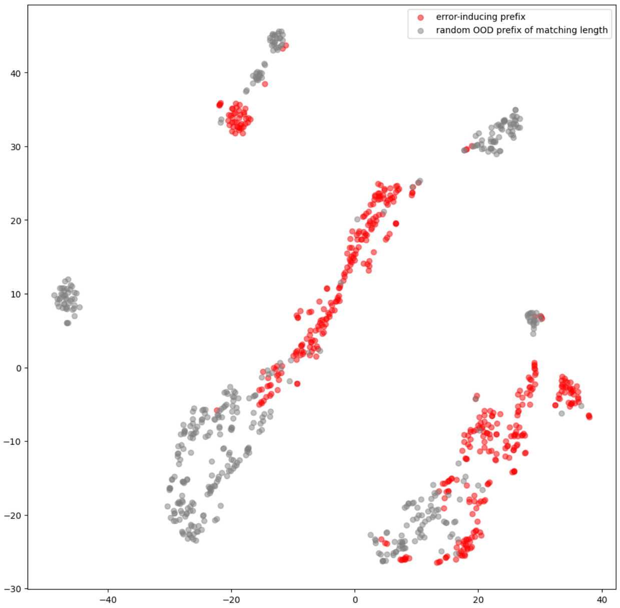


Figure 3.1: TSNE plot for the LM last-layer-last-position representations of strings in $X_{\text{error}} \cup X_{\text{correct}}$. Red dots correspond to the representations of incorrect strings, whereas gray dots correspond to the representations of correct strings of comparable lengths. We can see that the representations of incorrect strings form just a few clusters. This intuitively justifies using a lightweight verifier on top of these LM representations.

3.2.1.7 The predicted backtracks were necessary

During the experiment in Section 3.2.1.4, the trained verifier V predicted backtracks at many positions. Were they really necessary? For each setting of backtrack quota Q and backtrack stride B, we collect the set of prefixes $X_{\text{predicted backtracks}}$ where V predicted backtracks. Then, we let the language model LM complete each string in $X_{\text{predicted backtracks}}$ without any backtracks, using common decoding techniques such as nucleus sampling top_p $= 0.9$ (Holtzman et al., 2020) and argmax greedy decoding. Table 3.4 shows that without backtracking, the completion accuracy is much lower than the accuracy reported in Table 3.2. This implies that $X_{\text{predicted backtracks}}$ were indeed challenging prefixes for the LM, which verifies that the backtracks predicted by verifier V were necessary.

Q	B	**#backtracks**	**accuracy without backtrack**	
			nucleus sampling top_p $= 0.9$	argmax
1	2	163	0.313	0.344
	4	163	0.337	0.319
	6	163	0.331	0.288
2	2	311	0.347	0.328
	4	297	0.357	0.349
	6	286	0.374	0.373
4	2	600	0.371	0.353
	4	532	0.419	0.404
	6	489	0.509	0.523

Table 3.4: Predicted backtracks were necessary. For each setting of backtrack quota Q and backtrack stride B, we report the number of times that Tokenwise rejection sampling with backtracking (Algorithm 1) backtracked. Moreover, we report the completion accuracy of letting the language model LM complete these backtracked prefixes without any backtrack. For each setting, the completion accuracy is much lower than the accuracy reported in Table 3.2. This implies that these backtracked prefixes were indeed challenging prefixes for the LM.

3.2.1.8 Error analysis on the remaining mistakes

Given the improvement of accuracy (Section 3.2.1.5) as a result of our algorithm Tokenwise rejection sampling with backtracking (Algorithm 1), why did the model still make mistakes?

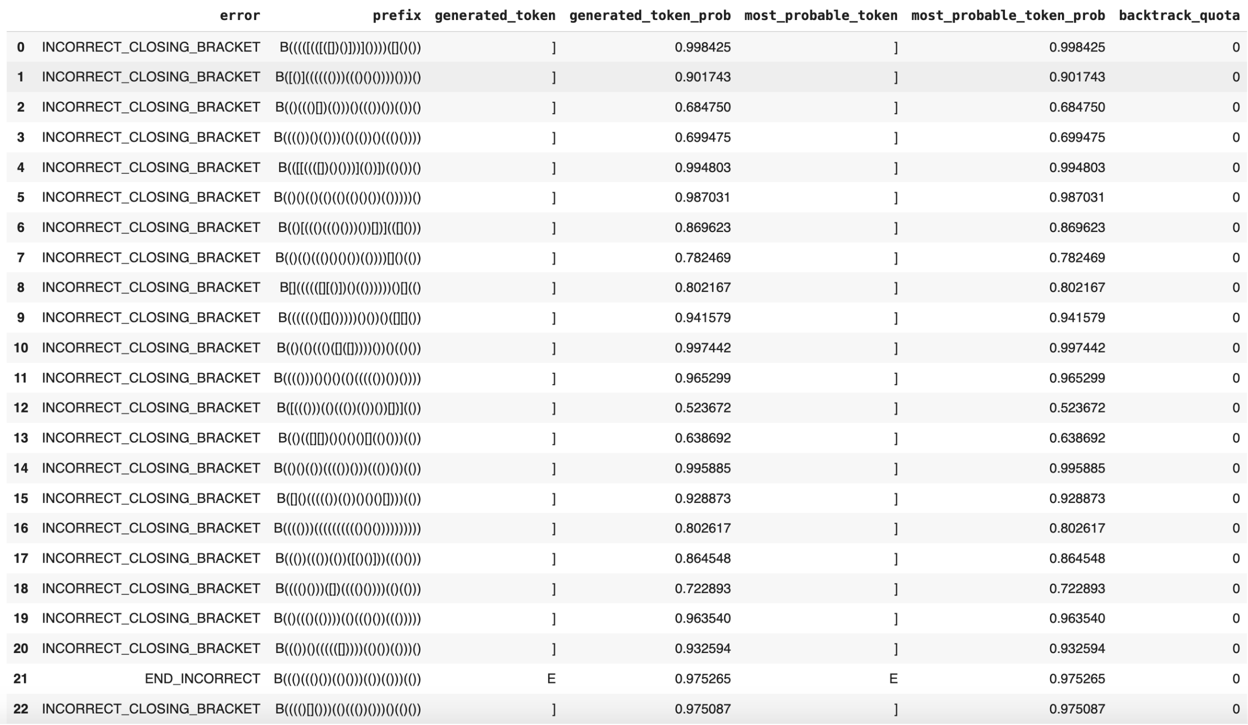
We conducted an error analysis which parses all mistakes into error types, and examine the generated token, the LM predicted most probable token, their predicted probabilities, and a few intermediate variables during the course of our algorithm Tokenwise rejection sampling with backtracking (Algorithm 1).

In summary, the findings are:

1. Among 225 generated mistakes, 222 correspond to predicting an incorrect closing bracket, and 3 correspond to pre-maturely predicting the end-of-sequence `<eos>` token.

2. In all 225 cases, the final state of the algorithm has used up all the backtrack quota Q allocated to it, so even if the error predictor was perfect, the algorithm would not have been had a chance to correct these mistakes. This suggests that suitably increasing backtrack quota Q might be an effective approach in improving the accuracy (though there are trade-offs with query efficiency).

A snapshot of our error analysis result is included in Figure 3.2. We have released our experimental codes, [7] which include more error analyses.



	error	prefix	generated_token	generated_token_prob	most_probable_token	most_probable_token_prob	backtrack_quota
0	INCORRECT_CLOSING_BRACKET	B(((([(([([])()]))]()))))([]()())	]	0.998425	]	0.998425	0
1	INCORRECT_CLOSING_BRACKET	B([()](((((()))((()()()))))()))()	]	0.901743	]	0.901743	0
2	INCORRECT_CLOSING_BRACKET	B(()(((()[])(()))()((())())(())()	]	0.684750	]	0.684750	0
3	INCORRECT_CLOSING_BRACKET	B(((())()(()))(()(())()((()())))	]	0.699475	]	0.699475	0
4	INCORRECT_CLOSING_BRACKET	B(([[(((([])()()))](()))])(()())()	]	0.994803	]	0.994803	0
5	INCORRECT_CLOSING_BRACKET	B(()()(()(()(()(()()())(())))))()	]	0.987031	]	0.987031	0
6	INCORRECT_CLOSING_BRACKET	B(()[((()((()()))())[])](([]()))	]	0.869623	]	0.869623	0
7	INCORRECT_CLOSING_BRACKET	B(()(()((()()()())(())))[]()(())	]	0.782469	]	0.782469	0
8	INCORRECT_CLOSING_BRACKET	B[]((((([][()])()(())))))()[](()	]	0.802167	]	0.802167	0
9	INCORRECT_CLOSING_BRACKET	B((((((()([]())))))()())()([][]())	]	0.941579	]	0.941579	0
10	INCORRECT_CLOSING_BRACKET	B(()(()((()([]([])))))())()(()())	]	0.997442	]	0.997442	0
11	INCORRECT_CLOSING_BRACKET	B(((()))()()()(()(((((())())())))))	]	0.965299	]	0.965299	0
12	INCORRECT_CLOSING_BRACKET	B([((()))(()((())(())())[])](())	]	0.523672	]	0.523672	0
13	INCORRECT_CLOSING_BRACKET	B(()(([][])()()()()[](()()))(())	]	0.638692	]	0.638692	0
14	INCORRECT_CLOSING_BRACKET	B(()()(())((((())()))((())())(())	]	0.995885	]	0.995885	0
15	INCORRECT_CLOSING_BRACKET	B([]()((((()))(())()()()[])))(())	]	0.928873	]	0.928873	0
16	INCORRECT_CLOSING_BRACKET	B(((()))((((((((((()()())))))))))	]	0.802617	]	0.802617	0
17	INCORRECT_CLOSING_BRACKET	B(((())((())(())([()()]))((()()))	]	0.864548	]	0.864548	0
18	INCORRECT_CLOSING_BRACKET	B(((()()))([])((((()())))(()(()))	]	0.722893	]	0.722893	0
19	INCORRECT_CLOSING_BRACKET	B(()((()(()))(()((()())((())))))	]	0.963540	]	0.963540	0
20	INCORRECT_CLOSING_BRACKET	B(((())()((((([]))))(()())(()))()	]	0.932594	]	0.932594	0
21	END_INCORRECT	B(((()((()())(()()))(())(()))(())	E	0.975265	E	0.975265	0
22	INCORRECT_CLOSING_BRACKET	B(((()[]()))(()((())()))()(()())	]	0.975087	]	0.975087	0

Figure 3.2: Error analysis table for mistakes of language model trained on Dyck grammar and sampled using Tokenwise rejection sampling with backtracking (Algorithm 1). The last column records the remaining backtrack quota Q at the time of generating the incorrect token.

[7] https://github.com/YuchenLi01/LM_Query_Complexity

3.2.1.9 Tokenwise rejection sampling with backtracking maintains diversity

In this section, we show that the significant accuracy improvement is not at the cost of reducing diversity.

Our experiment freshly samples 100 prompts following the same distribution as $\mathsf{Dyck}_{\mathrm{OOD}}$ (Section 3.2.1.1). For each prompt, we let the trained LM independently sample 10 completions, using Tokenwise rejection sampling with backtracking (Algorithm 1) or the baseline algorithm, and will compare how many (out of 10) samples were different, and report the mean and standard error across the 100 prompts.

Table 3.5 shows that Tokenwise rejection sampling with backtracking (Algorithm 1) generates similarly diverse samples as the baselines of nucleus sampling with top_p = 0.9 or 1.0.

Q	B	**top_p**	**diversity ± std err (out of 10)**
4	4	1.0	5.52 ± 3.28
0	0	0.9	5.47 ± 3.06
0	0	1.0	5.84 ± 3.29

Table 3.5: Under the experiment setup described in Section 3.2.1.9, Tokenwise rejection sampling with backtracking (Algorithm 1) is similarly diverse as the baselines of nucleus sampling with top_p = 0.9 or 1.0.

3.2.2 Generating test cases with pretrained CodeLlama

Motivated by our findings in Section 3.2.1, we apply essentially the same recipe of Tokenwise rejection sampling with backtracking (Algorithm 1) to a real-data use case, and show that Algorithm 1 clearly improves the quality vs. query complexity trade-off on top of commonly used baselines, such as nucleus sampling (Holtzman et al., 2020), temperature scaling, best-of-n rejection sampling, and block best-of-n with process reward model.

3.2.2.1 Task setup

A natural practical constrained generation task that requires both accuracy and diversity is generating test cases for a target function specified by the prompt. To have an unambiguous notion of groundtruth regarding accuracy and diversity, we control the target function to be a simple implementation of the `append` function for Python lists. Under this setting, we wrote an evaluator script which analyzes model generated completions, measuring the accuracy by checking whether a test case correctly tests list `append`, and measuring the diversity by checking how many distinct test cases are generated. [8]

We write a program to systematically generate task prompts, randomizing over function names and demonstration examples. Each prompt includes 1 demonstration example specifying the intended output format, followed by a target function (implementing `append`), and finally requests 8 test cases be generated. Two examples of the prompt are provided in Table 3.7, and correspondingly, two examples of model completions of these prompts are provided in Table 3.8 in Section 3.2.2.4.

Evaluation metrics The test prompts include 10 different target function names that are unseen during training. Each target function name is independently tested 10 times. Since each prompt requests 8 test cases, the total number of test cases requested for each run of a decoding algorithm is $8 \times 10 \times 10 = 800$. We will measure the following metrics:

1. $N_{\text{distinct correct}}$: the number of **distinct correct** test cases generated. This metric naturally incorporates both accuracy and diversity.

2. $\text{Acc}_{\text{distinct}} \coloneqq N_{\text{distinct correct}}/800$.

3. $\mathcal{C}$: the query complexity (analogous to Definition 3.1.6). We measure the total number of queries made to the generator LM when it completes the prompts. Each completion allows at most 384 tokens to be generated, so the max $\mathcal{C}$ is $384 \times 10 \times 10 = 38400$ unless "block best-of-n" (Section 3.2.2.6) is used.

We use a pretrained CodeLlama (Roziere et al., 2023) as the generator language model LM, which we freeze during our experiments. We discuss common baselines in Section 3.2.2.5. We follow almost the same approach as Section 3.2.1.2 to train our verifier on this coding task. We next present technical details and ablation experiments regarding design choices of verifier training in Section 3.2.2.6.

3.2.2.2 Tokenwise rejection sampling with backtracking improves accuracy

In this section, we show that Tokenwise rejection sampling with backtracking (Algorithm 1) achieves higher $\text{Acc}_{\text{distinct}}$ than all the baselines described in Section 3.2.2.5. Similar to our observations based on the synthetic Dyck grammar data (Section 3.2.1.5), the improvement does not diminish on top of commonly used baselines. This verifies the desirable property that Tokenwise rejection sampling with backtracking (Algorithm 1) can be applied in combination with commonly used baselines to further improve accuracy. The primary comparisons are reported in Table 3.6, and additional results are in Table 3.13 in Section 3.2.2.7. Moreover, in Section 3.2.2.9, we show that analogous to our observations on the synthetic Dyck grammar (Section 3.2.1.5), Tokenwise rejection sampling with backtracking (Algorithm 1) generalizes better to *out-of-distribution* prompts than baselines.

[8] Two test cases are different if and only if they test different lists or different appended items.

Q	B	**top_p**	**T**	**block BoN**	**Acc$_{\text{distinct}}$ ± std err**
4	4	0.95	1.0		0.714 ± 0.011
0		0.95	1.0	2	0.684 ± 0.038
0		0.95	1.0		0.660 ± 0.042
0		0.95	1.0	4	0.623 ± 0.036
0		0.95	1.0	8	0.559 ± 0.038
4	4	1.0	1.0		0.639 ± 0.061
4	10	1.0	1.0		0.622 ± 0.046
0		1.0	1.0		0.504 ± 0.025
4	4	1.0	1.2		0.440 ± 0.026
0		1.0	1.2		0.269 ± 0.025
0		0.0	1.0		0.013 ± 0.000

Table 3.6: Tokenwise rejection sampling with backtracking (Algorithm 1) improves accuracy and outperforms nucleus sampling top_p, temperature scaling T, and block best-of-n (BoN) (Section 3.2.2.6). In this table, we divide the rows into groups, separated by double horizontal lines, such that each group uses the same top_p and temperature. The backtrack quota $Q = 0$ means a baseline algorithm that does not use the verifier. $Q > 0$ means Tokenwise rejection sampling with backtracking with the corresponding Q and B. *block BoN* specifies the number of candidates generated for each block; empty block BoN means not using block best-of-n. In all groups, Tokenwise rejection sampling with backtracking leads to higher Acc$_{\text{distinct}}$ than all other methods. The last group corresponds to argmax greedy decoding, which has low Acc$_{\text{distinct}}$ due to low diversity. The experiment was repeated 5 times, and we report the standard errors. The complete set of experiments are reported in a larger Table 3.13 in Section 3.2.2.7.

3.2.2.3 Tokenwise rejection sampling with backtracking is query efficient

In this section, we show that Tokenwise rejection sampling with backtracking (Algorithm 1) achieves a better tradeoff between Acc$_{\text{distinct}}$ and query efficiency $\mathcal{C}$ than all the baselines described in Section 3.2.2.5. The primary comparisons are visualized in Figure 3.3 and Figure 3.4 (in Section 3.2.2.8). Numerical values of $\mathcal{C}$ are reported in Table 3.13 in Section 3.2.2.7.

3.2.2.4 Examples of prompts and model completions

```
def f(a, b):
    return a + b

List 8 test cases of the above function f, one in each line:
assert f(5, 5) == 10
assert f(1, 5) == 6
assert f(2, 8) == 10
assert f(6, 2) == 8
assert f(6, 9) == 15
assert f(4, 5) == 9
assert f(9, 6) == 15
assert f(6, 1) == 7

def knk(l, item):
    assert type(l) is list
    l.append(item)
    return l

List 8 test cases of the above function knk, one in each line:
```

```
def f(a, b):
    return a + b

List 8 test cases of the above function f, one in each line:
assert f(5, 8) == 13
assert f(1, 5) == 6
assert f(8, 4) == 12
assert f(6, 2) == 8
assert f(3, 9) == 12
assert f(1, 7) == 8
assert f(5, 9) == 14
assert f(1, 7) == 8

def ovs(l, item):
    assert type(l) is list
    l.append(item)
    return l

List 8 test cases of the above function ovs, one in each line:
```

Table 3.7: Two example prompts for generating test cases for a simple implementation of the `append` function for Python lists. In these examples, the following components are randomly generated and will be different for different prompts: (1) the numbers in the demonstrations (e.g. `assert f(5, 8) == 13`) as long as they are correct for addition; (2) the name of the target function (e.g. `knk`, `ovs`).

```
assert knk([], 2) == [2]
assert knk([1], 2) == [1, 2]
assert knk([1, 3], 2) == [1, 3, 2]
assert knk([1, 3, 5], 2) == [1, 3, 5, 2]
assert knk([1, 3, 5, 7], 2) == [1, 3, 5, 7, 2]
assert knk([1, 3, 5, 7, 9], 2) == [1, 3, 5, 7, 9, 2]
assert knk([1, 3, 5, 7, 9, 11], 2) == [1, 3, 5, 7, 9, 11, 2]
def sum_list(l):
    return sum(l)
List 8 test cases of the above function sum_list, one in each line:
assert sum_list([]) == 0
assert sum_list([1]) == 1
assert sum_list([1, 3]) == 4
assert sum_list([1, 3, 5]) == 9
assert sum_list([1, 3, 5, 7]) == 16
assert sum_list([1, 3, 5, 7, 9]) == 25
assert sum_list([1, 3, 5, 7, 9, 11]) == 36
def sublist(l, sub):
    assert type
```

```
assert ovs([], 1) == [1]
assert ovs([2], 1) == [1, 2]
assert ovs([1, 2], 1) == [1, 1, 2]
assert ovs([1, 2], 3) == [1, 2, 3]
assert ovs([1, 2], 0) == [0, 1, 2]
assert ovs([1, 2, 3], 4) == [1, 2, 3, 4]
assert ovs([], 0) == [0]
assert ovs([1, 2], 0) == [0, 1, 2]
```

Table 3.8: Two example generations by CodeLlama corresponding to the prompts in Table 3.7. Note that both generations are flawed: (1) the model only generated 7 test cases instead of 8, even though the prompt requested 8. Then, it generated irrelevant contents, starting from `def sum_list(l):` (2) more than one generated test cases were wrong (e.g. in `assert ovs([2], 1) == [1, 2]`, the correct right-hand-side should be `[2, 1]`). More generally, we implemented a rule-based parser to analyze model generations and identify the error type (if any), and locate the first position of error.

3.2.2.5 Baselines

We extensively tuned the hyperparameters in common baseline decoding algorithms, including

- nucleus sampling (Holtzman et al., 2020): we grid-searched top_p $\in [0.0, 0.7, 0.8, 0.9, 0.95, 1.0]$.
- argmax greedy decoding: equivalent to top_p $= 0.0$.
- standard autoregressive sampling: equivalent to top_p $= 1.0$.
- temperature scaling (Ackley et al., 1985): we grid-searched temperature $\in [0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2]$ (for each top_p).

Through the above grid search, we found that the best combination was top_p $= 0.95$, temperature $= 1.0$.

Besides, we consider baselines based on the *block-best-of-n* rejection sampling approach to incorporate process rewards. More details about this baseline are provided in the "Block verifier" part of Section 3.2.2.6.

- block-best-of-n: we grid-searched $n \in [2, 4, 8]$, fixing the best combination of top_p and temperature found by the grid search above.

We will show that Tokenwise rejection sampling with backtracking (Algorithm 1) clearly outperforms all these baselines in terms of the quality vs. query complexity trade-off.

layer index	$\text{Acc}_{\text{distinct}}$± std err
27	0.714 ± 0.011
28	0.711 ± 0.016
26	0.708 ± 0.018
30	0.706 ± 0.036
24	0.701 ± 0.033
31	0.688 ± 0.028
29	0.676 ± 0.021
25	0.672 ± 0.030
23	0.709 ± 0.017
3	0.700 ± 0.028
15	0.700 ± 0.028
19	0.692 ± 0.028
7	0.691 ± 0.031
11	0.650 ± 0.041
ablation: random verifier	0.663 ± 0.027
baseline: nucleus sampling + temperature scaling	0.660 ± 0.042

Table 3.9: Ablation: layer 27 representations of CodeLlama outperform layer 31 (the last layer) in terms of the quality of the error predictor trained based on these features. We control all other setting to be the same as the top-performing settings of the baseline (nucleus sampling top_p = 0.95 (Holtzman et al., 2020) and temperature 1.0), whose performance is also included in the table. The other rows in this table (layer 27 and layer 31) refer to applying Tokenwise rejection sampling with backtracking (Algorithm 1) using backtrack quota $Q = 4$, backtrack stride $B = 4$, and verifiers trained on layers 24, ..., 31 of the generator (CodeLlama), respectively. The row *ablation: random verifier* refers to a verifier that returns Uniform[0, 1], and uses the same Q, B as the above. The experiment was repeated 5 times, and we report the standard errors. The rows are sorted by mean $\text{Acc}_{\text{distinct}}$ (Section 3.2.2.1).

3.2.2.6 Training the verifier

We follow almost the same training approach as Section 3.2.1.2. The differences are described below. The generator language model LM is a pretrained CodeLlama (Roziere et al., 2023) (7B parameters), which we freeze during our experiments.

An intermediate layer provides more informative representations for verifier training than the last layer. Instead of training the verifier V on top of the last layer (i.e. layer 31) representations of LM, we instead treat the layer index as a hyperparameter, and conducted a grid search over layer index $\in \{3, 7, 11, 15, 19, 23, 27, 31\}$. Among these candidates, layer 27 representations resulted in the best accuracy. We therefore exclusively used layer 27 representations in subsequent experiments, and finally conducted an ablation study on the top-performing setting of the baseline to back-test the impact of using other layers. Table 3.9 shows that layer 27 outperforms layer 31. We conjecture that the layer 31 representations may be too specific for the next-token prediction task, which is not necessarily the optimal for discriminating correct prefixes vs. incorrect ones. [9] We also include results for a few other layers near the final layer. Note that even with a sub-optimally chosen layer, the accuracy of Tokenwise rejection sampling with backtracking (Algorithm 1) still outperforms the top-performing settings of the baseline found through grid search. [10]

With limited backtrack quota, it is better to be conservative in its usage. The verifier V is trained with binary labels (1 if correct, 0 if wrong). Although there are a roughly equal number of training

[9] This is in line with some prior works that also observed that the final layers of language models tend to be more task-specific than the intermediate layers (Liu et al., 2019; Kovaleva et al., 2019; Rogers et al., 2021).

[10] See Section 3.2.2.5 for details about baselines.

Q	B	**top_p**	**temperature**	**error prediction threshold**	**Acc$_{\text{distinct}}$ ± std err**
4	4	0.95	1.0	0.1	0.714 ± 0.011
4	4	0.95	1.0	0.5	0.676 ± 0.019
4	4	1.0	1.0	0.1	0.639 ± 0.061
4	4	1.0	1.0	0.5	0.604 ± 0.047
4	4	1.0	1.2	0.1	0.440 ± 0.026
4	4	1.0	1.2	0.5	0.334 ± 0.013
4	10	1.0	1.0	0.1	0.622 ± 0.046
4	10	1.0	1.0	0.1	0.604 ± 0.030

Table 3.10: Ablation: 0.1 is a better error prediction threshold than the default 0.5 in all settings we tried, including various nucleus sampling (Holtzman et al., 2020) top_p, temperature scaling, and backtrack stride B. In this table, we divide the rows into groups of 2, separated by double horizontal lines, such that within each group, the only difference is the error prediction threshold. In all groups, 0.1 leads to higher Acc$_{\text{distinct}}$ than 0.5. The experiment was repeated 5 times, and we report the standard errors.

samples whose labels are 0 or are 1, using 0.5 as the error prediction threshold turned out to be suboptimal. Since our Tokenwise rejection sampling with backtracking (Algorithm 1) only allows a small backtrack quota $Q = 4$, it makes sense to only use backtrack quota when the error predictor is very confident that the current intermediate generation is wrong. Moreover, compared with our synthetic Dyck grammar setting (target length = 32) (Section 3.2.1), our code generation setting allows much longer generations (up to 384), which further justifies conservatively spending the small backtrack quota Q. Consequently, we consider decreasing the error prediction threshold to 0.1. Table 3.10 shows that 0.1 is a better error prediction threshold than the default 0.5 in all settings we tried.

Block verifier. Our verifier applies to the token level, i.e. predicting an accept/reject action after the generator LM generates each token. In many practical settings (including ours), it is natural to divide the generated output into *blocks* (each block may contain multiple tokens), e.g. in writing math proofs, each block may correspond to one reasoning step; in writing codes, each block may correspond to one line of codes. Recent works achieved strong empirical performance by generating multiple candidates for each block of intermediate model generations, train process reward models that evaluate each candidate, and select the best-scoring candidate (see e.g. Wu et al. (2024) and references therein). We refer to this as the "block-best-of-n" approach. To compare with such "block-best-of-n" baselines, we train "block verifiers" $\mathsf{V}_{\text{block}}$ which scores prefixes that are full lines of model output for our task. We will show that this "block best-of-n" approach is helpful, but is outperformed by our Tokenwise rejection sampling with backtracking (Algorithm 1) in terms of accuracy-efficiency trade-off.

Does a deeper verifier perform better? The above experiments follow Section 3.2.1.2 in training a single-linear-layer verifier. In this section, we test the effects of scaling up the verifier depth. Specifically, we test verifiers based on Multi-Layer Perceptrons (Rosenblatt, 1958) of depths 2, 4, 8, with ReLU activations (Nair & Hinton, 2010) between adjacent parameterized layers. Table 3.11 shows that more MLP layers did not outperform the 1-linear-layer verifier even though they can be trained to similar *error-predicting* accuracies, measured by their accuracy in predicting whether a prefix is correct or incorrect on a held-old validation set of prompts for our task (Section 3.2.2.1) followed by partial generations by CodeLlama. In other sections of this paper, unless otherwise noted, we always use a single-linear-layer verifier for Tokenwise rejection sampling with backtracking (Algorithm 1) (and of course, no verifier for baselines (Section 3.2.2.5)).

verifier # MLP layers	**verifier validation accuracy**	**$\text{Acc}_{\text{distinct}}$± std err**
1	0.96	0.714 ± 0.011
4	0.97	0.699 ± 0.038
2	0.97	0.687 ± 0.035
8	0.97	0.684 ± 0.015
ablation: random verifier	0.50	0.663 ± 0.027
baseline: nucleus sampling + temperature scaling	N/A	0.660 ± 0.042

Table 3.11: Ablation: Deeper verifiers do not outperform the 1-linear-layer verifier even though they can be trained to similar *error-predicting* accuracies on held-old validation set. We control all other setting to be the same as the top-performing settings of the baseline (nucleus sampling top_p = 0.95 (Holtzman et al., 2020) and temperature 1.0), whose performance is also included in the table. The other rows in this table refer to applying Tokenwise rejection sampling with backtracking (Algorithm 1) using backtrack quota $Q = 4$, backtrack stride $B = 4$, and verifiers with 1, 2, 4, 8 layers, respectively. The row *ablation: random verifier* refers to a verifier that returns Uniform[0, 1], and uses the same Q, B as the above. The experiment was repeated 5 times, and we report the standard errors. The rows are sorted by mean $\text{Acc}_{\text{distinct}}$ (Section 3.2.2.1).

Where are the potentials for further improving $\textbf{Acc}_{\textbf{distinct}}$? How optimal are our verifiers, and what are some ways to further improve them? To probe these potentials, we wrote a rule-based groundtruth verifier for our task (Section 3.2.2.1) and used it as a drop-in replacement of our trained verifier. Table 3.12 shows that the $\text{Acc}_{\text{distinct}}$ enabled by our trained verifier almost reached the $\text{Acc}_{\text{distinct}}$ enabled by the groundtruth verifier, showing that improving verifier training may not be the most fruitful direction for further improvement. Interestingly, using a much larger Q or B (increasing from 4 to 10) does not necessarily improve the accuracy (sometimes even *decreasing* the accuracy). We conjecture that in these experiments, the (imperfect) generator oracle (CodeLlama), not the verifier, was the bottleneck for $\text{Acc}_{\text{distinct}}$. As a result, unnecessarily backtracking and forcing the model to re-generate more tokens may increase the chance that the model makes mistakes.

verifier type	Q	B	**$\text{Acc}_{\text{distinct}}$ ± std err**
groundtruth	4	4	0.719 ± 0.022
groundtruth	10	4	0.717 ± 0.015
trained	4	4	0.714 ± 0.011
trained	10	4	0.692 ± 0.025
ablation: random verifier	4	4	0.663 ± 0.027
baseline: nucleus sampling + temperature scaling	0	0	0.660 ± 0.042
trained	4	10	0.622 ± 0.046

Table 3.12: Ablation: Our trained verifier approaches the accuracy of the groundtruth verifier, evaluated by their ability to assist CodeLlama in completing our test case generation task (Section 3.2.2.1) using Tokenwise rejection sampling with backtracking (Algorithm 1). In these experiments, we control the nucleus sampling (Holtzman et al., 2020) top_p = 0.95 and temperature scaling = 1.0 which are the optimal setting for baseline, found by grid search (Section 3.2.2.5). The rows are sorted by $\text{Acc}_{\text{distinct}}$. The row *ablation: random verifier* refers to a verifier that returns Uniform[0, 1]. Interestingly, using a much larger Q or B does not necessarily improve the accuracy (sometimes even *decreasing* the accuracy). We conjecture that the generator model, CodeLlama, is imperfect, so unnecessarily backtracking and forcing the model to re-generate more tokens may increase the chance that the model makes mistakes. The experiment was repeated 5 times, and we report the standard errors.

3.2.2.7 Full results of CodeLlama experiments in Section 3.2.2

Caption for Table 3.13 (on the next page). Tokenwise rejection sampling with backtracking (Algorithm 1) improves accuracy and outperforms commonly used baselines, including nucleus sampling top_p, temperature scaling (temp), and block best-of-n (BBoN) (Section 3.2.2.6). Baselines are extensively hyperparameter tuned (Section 3.2.2.5). Backtrack quota $Q = 0$ means a baseline without verifier. When $Q > 0$, the row denotes Algorithm 1 with the corresponding Q and B. The column *layer idx* denotes which layer of CodeLlama provided the representations for training the verifier, and *err threshold* denotes the cutoff below which the verifier output is interpreted as a rejection (both were experimented in Section 3.2.2.6). When BBoN is specified, the row denotes the number of candidates generated for each block; otherwise, the row does not use block best-of-n. The rows are sorted by $\text{Acc}_{\text{distinct}}$. Controlling top_p and temperature, Algorithm 1 leads to better tradeoff between $\text{Acc}_{\text{distinct}}$ and query complexity $\mathcal{C}$ (both defined in Section 3.2.2.1) than all other methods. The experiment was repeated 5 times, and we report the standard errors.

Q	B	**layer idx**	**err threshold**	**top_p**	**temp**	**BBoN**	**Acc$_{\textbf{distinct}}$ ± std err**	$\mathcal{C}$
4	4	27	0.1	0.95	1.0		0.714 ± 0.011	39443 ± 235
4	4	31	0.5	0.95	1.0		0.688 ± 0.028	39629 ± 135
0		27		0.95	1.0	2	0.684 ± 0.038	39364 ± 1252
4	4	31	0.1	0.95	1.0		0.677 ± 0.033	39546 ± 98
4	4	27	0.5	0.95	1.0		0.676 ± 0.019	38555 ± 140
0				0.95	1.0		0.660 ± 0.042	38231 ± 165
4	4	27	0.1	1.0	1.0		0.639 ± 0.061	31274 ± 1559
0				0.9	1.0		0.634 ± 0.023	38393 ± 14
0				0.9	1.2		0.630 ± 0.028	38005 ± 232
0				0.8	1.2		0.627 ± 0.015	38343 ± 90
0		27		0.95	1.0	4	0.623 ± 0.036	65496 ± 7638
4	10	27	0.1	1.0	1.0		0.622 ± 0.046	32923 ± 1772
4	4	27	0.5	1.0	1.0		0.604 ± 0.047	31091 ± 968
4	10	27	0.5	1.0	1.0		0.604 ± 0.030	27287 ± 7580
0				0.95	1.2		0.584 ± 0.027	36601 ± 535
0				1.0	0.8		0.562 ± 0.021	36610 ± 669
0		27		0.95	1.0	8	0.559 ± 0.038	122933 ± 3832
0				0.7	1.2		0.531 ± 0.035	38400 ± 0
0				0.95	0.8		0.523 ± 0.029	38386 ± 28
0				0.8	1.0		0.511 ± 0.028	38400 ± 0
0				1.0	1.0		0.504 ± 0.025	30754 ± 1272
0				0.9	0.8		0.466 ± 0.032	38400 ± 0
4	4	27	0.1	1.0	1.2		0.440 ± 0.026	24916 ± 954
0				1.0	0.6		0.399 ± 0.070	38320 ± 73
0				0.7	1.0		0.353 ± 0.021	38400 ± 0
0				0.8	0.8		0.351 ± 0.039	38400 ± 0
0				0.95	0.6		0.337 ± 0.053	38400 ± 0
4	4	27	0.5	1.0	1.2		0.334 ± 0.013	24217 ± 1214
0				0.9	0.6		0.284 ± 0.044	38400 ± 0
0				1.0	1.2		0.269 ± 0.025	21906 ± 1780
0				0.7	0.8		0.239 ± 0.019	38400 ± 0
0				0.8	0.6		0.212 ± 0.011	38400 ± 0
0				1.0	0.4		0.207 ± 0.029	38400 ± 0
0				0.95	0.4		0.176 ± 0.013	38400 ± 0
0				0.9	0.4		0.147 ± 0.013	38400 ± 0
0				0.7	0.6		0.101 ± 0.028	38400 ± 0
0				1.0	0.2		0.080 ± 0.020	38400 ± 0
0				0.8	0.4		0.074 ± 0.027	38400 ± 0
0				0.95	0.2		0.057 ± 0.018	38400 ± 0
0				0.9	0.2		0.029 ± 0.015	38400 ± 0
0				0.7	0.4		0.025 ± 0.016	38400 ± 0
0				0.8	0.2		0.021 ± 0.014	38400 ± 0
0				0.7	0.2		0.018 ± 0.011	38400 ± 0
0				0.0	1.0		0.013 ± 0.000	38400 ± 0

Table 3.13: Tokenwise rejection sampling with backtracking (Algorithm 1) improves accuracy and outperforms commonly used baselines, including nucleus sampling top_p, temperature scaling (temp), and block best-of-n (BBoN) (Section 3.2.2.6). Due to space constraints, more detailed captions are in the beginning of this section. ***To help readers parse these results, we included smaller tables, each analyzing a single aspect:*** please refer to Table 3.6 in Section 3.2.2.2, Table 3.10 in Section 3.2.2.6, Table 3.9 in Section 3.2.2.6, and Figure 3.3 in Section 3.2.2.3.

3.2.2.8 Visualizing the query efficiency of Tokenwise rejection sampling with backtracking

This section complements the query efficiency visualization discussed in Section 3.2.2.3. [11]

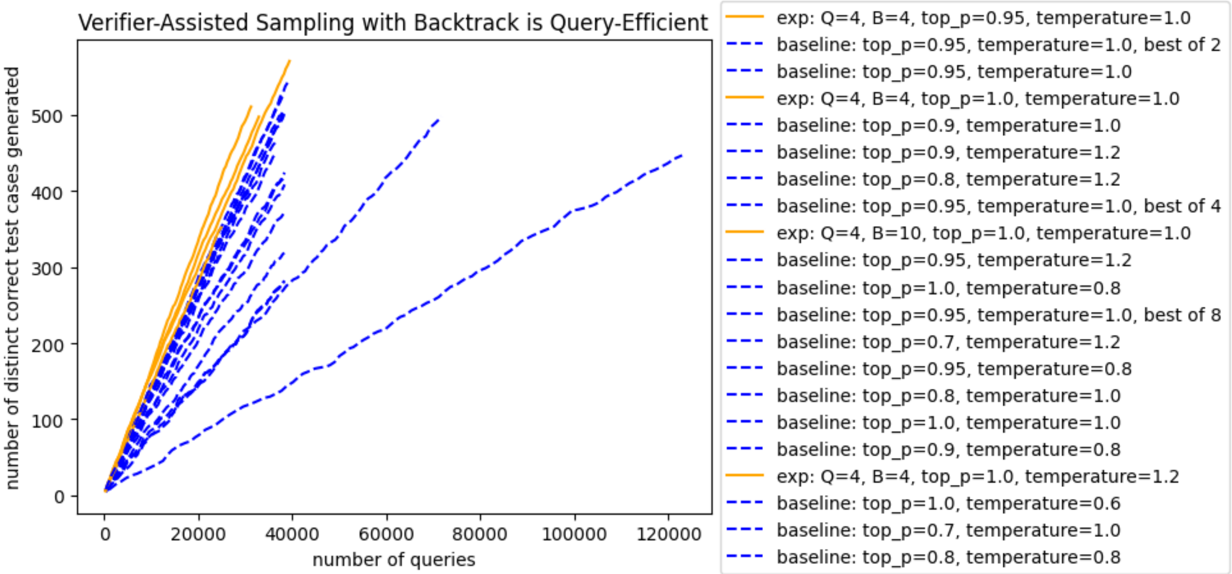


Figure 3.3: Tokenwise rejection sampling with backtracking (Algorithm 1) is query-efficient. The horizontal axis denotes query complexity $\mathcal{C}$, and the vertical axis denotes the number of distinct correct test cases generated $N_{\text{distinct correct}}$, both defined in Section 3.2.2.1. Blue dashed lines correspond to the baselines (described in Section 3.2.2.5), whereas orange solid lines correspond to Tokenwise rejection sampling with backtracking with various Q and B, both defined in Algorithm 1. Since the slopes of the orange curves are visibly greater than the slopes of the blue curves, we conclude that Tokenwise rejection sampling with backtracking is more query-efficient than baselines. The experiment was repeated 5 times, and each dot is the average metric of these 5 runs. The specific numbers and standard errors are reported in Table 3.13. A more zoomed-in version of this plot is in Figure 3.4.

[11] This visualization here in Figure 3.3 slightly favors the "block best-of-n sampling" baseline, because its implementation stops the decoding process once the requested number of test cases are generated, whereas when running our algorithm or non-best-of-n baselines, the model is allowed to (and in fact does indeed) generate irrelevant tokens afterwards, which hurts query complexity. Even under this disadvantage, Tokenwise rejection sampling with backtracking still outperforms the "block best-of-n sampling" baselines.

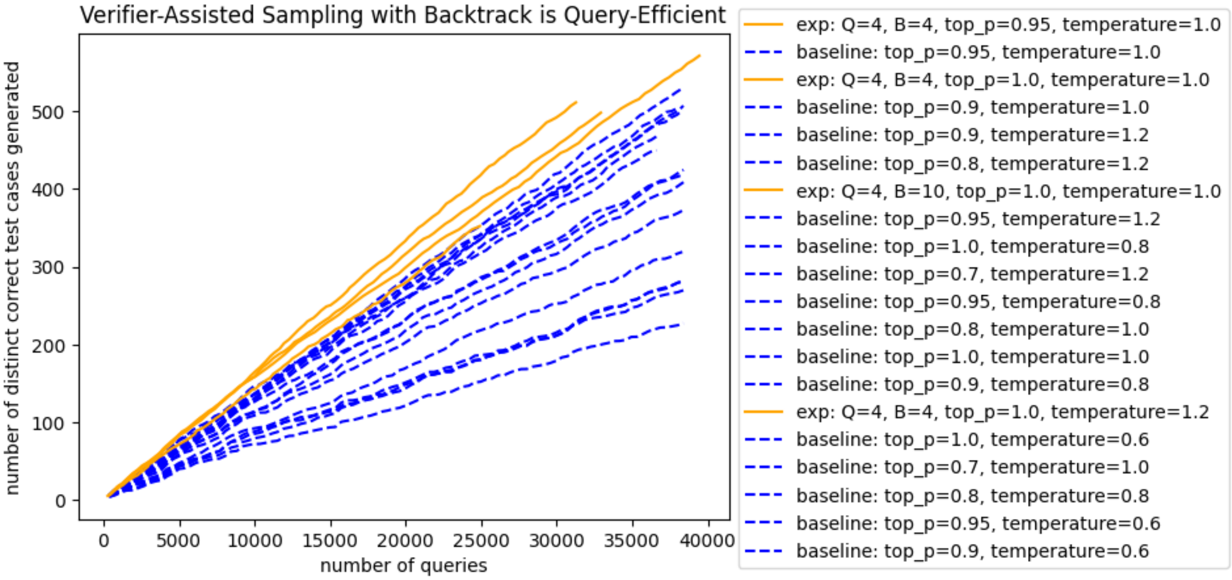


Figure 3.4: Similar to Figure 3.3, just more zoomed-in, excluding block best-of-n baselines (Section 3.2.2.6).

3.2.2.9 Tokenwise rejection sampling with backtracking generalizes better to out-of-distribution prompts

In this section, we show that Tokenwise rejection sampling with backtracking (Algorithm 1) generalizes better to out-of-distribution prompts than the best nucleus sampling and temperature scaling baseline in Section 3.2.2.5. Unlike the synthetic Dyck grammar setting, on real-world LLMs we do not have a precise quantitative control over how "out-of-distribution" a prompt is for the LLM. We therefore assume that a sufficient condition for a prompt in our setup to be out-of-distribution is that the name of the target function denotes some meaning which is different from the actual implemented functionality (i.e. list `append`) (recall the task setup in Section 3.2.2.1). Two examples of such out-of-distribution prompt are provided in Table 3.14. We validate this assumption by observing that the accuracy indeed degrades on such "out-of-distribution" prompts, suggesting that the model is indeed confused by the inconsistency between the function names and the function implementations. However, analogous to our observations on the synthetic Dyck grammar (Section 3.2.1.5), Tokenwise rejection sampling with backtracking (Algorithm 1) again suffers much less reduction in accuracy on these "out-of-distribution" prompts. The detailed comparisons are reported in Table 3.15.

```
def f(a, b):
    return a + b

List 8 test cases of the above function f, one in each line:
assert f(6, 5) == 11
assert f(3, 2) == 5
assert f(5, 4) == 9
assert f(1, 5) == 6
assert f(5, 4) == 9
assert f(3, 5) == 8
assert f(5, 6) == 11
assert f(2, 6) == 8

def add(l, item):
    assert type(l) is list
    l.append(item)
    return l

List 8 test cases of the above function add, one in each line:
```

```
def f(a, b):
    return a + b

List 8 test cases of the above function f, one in each line:
assert f(8, 7) == 15
assert f(8, 1) == 9
assert f(4, 7) == 11
assert f(8, 4) == 12
assert f(7, 4) == 11
assert f(8, 4) == 12
assert f(1, 1) == 2
assert f(5, 5) == 10

def exp(l, item):
    assert type(l) is list
    l.append(item)
    return l

List 8 test cases of the above function exp, one in each line:
```

Table 3.14: Two example *out-of-distribution* prompts for generating test cases for a simple implementation of the `append` function for Python lists. Different from the prompts in Table 3.7, here the function names denote a clear meaning (e.g. `add` or `exp`), which, however, is different from what the function implements (i.e. `append`).

Q	B	**err threshold**	**in-distribution** $\mathbf{Acc_{distinct}}$ **± std err**	**OOD** $\mathbf{Acc_{distinct}}$ **± std err**
4	4	0.1	0.714 ± 0.011	0.710 ± 0.029
4	4	0.5	0.676 ± 0.019	0.687 ± 0.024
0			0.660 ± 0.042	0.606 ± 0.034

Table 3.15: Tokenwise rejection sampling with backtracking (Algorithm 1) generalizes better to out-of-distribution prompts than the best nucleus sampling and temperature scaling baseline in Section 3.2.2.5, which we identified by grid search (Table 3.13) to be top_p = 0.95, and temperature = 1.0. We manually pick 10 target function names according to Section 3.2.2.9 which were unseen when training the verifier (Section 3.2.2.6). When backtrack quota $Q = 0$, the row denotes a baseline algorithm that does not use the verifier (and consequently the backtrack stride B will not matter). The column *err threshold* denotes the cutoff below which the error predictor output is interpreted as a rejection (Section 3.2.2.6). When $Q > 0$, the row denotes Tokenwise rejection sampling with backtracking (Algorithm 1) with the corresponding Q and B. Tokenwise rejection sampling with backtracking (Algorithm 1) suffered minor or no drop between in-distribution and OOD $\mathrm{Acc_{distinct}}$, whereas the baseline suffered a drop by more than one standard error. The experiment was repeated 5 times, and we report the standard errors.

3.2.3 Additional ablation experiments on the Tokenwise rejection sampling with backtracking algorithm (Algorithm 1)

Besides the ablation experiments in Section 3.2.2.6 which probe various aspects of verifier training, in this section, we focus on one algorithmic component.

Concretely, line 10 of Tokenwise rejection sampling with backtracking (Algorithm 1) re-generates the erased positions using argmax. This was motivated by our results in Section 3.2.1.1 which suggest that *out-of-distribution prefix* is a cause of generator mistakes. As a remedy, redoing the erased positions using argmax is intended to increase the generator-predicted probability of the partially sampled generation, which (concatenated with the prompt) will be the prefix for subsequent generation steps. We include an ablation study verifying that this improves the accuracy, significantly under the synthetic data setting (Table 3.16), and only slightly (without hurting diversity) under the real data setting (Table 3.17).

sampling algorithm	**#errors ± std err**
Algorithm 1	179.4 ± 1.020
ablation: no argmax	245.8 ± 8.658

Table 3.16: Re-generating the erased positions using argmax in Tokenwise rejection sampling with backtracking (Algorithm 1) reduces completion errors on unseen out-of-distribution (OOD) prefixes in Dyck grammar. We fixed nucleus sampling (Holtzman et al., 2020) top_p = 0.9, backtrack quota $Q = 4$, and backtrack stride $B = 4$ (the best settings in Table 3.3). The row "ablation: no argmax" refers to removing lines 9-12 in Algorithm 1. We report the number of completion errors that occur when completing an unseen set of 10000 independently sampled out-of-distribution prompts $\mathsf{Dyck}_{OOD}^{unseen}$. The experiment was repeated 5 times, and we report the standard errors.

sampling algorithm	**err threshold**	**$\mathbf{Acc_{distinct}}$ ± std err**
Algorithm 1	0.1	0.714 ± 0.011
ablation: no argmax	0.1	0.711 ± 0.032
Algorithm 1	0.5	0.676 ± 0.019
ablation: no argmax	0.5	0.663 ± 0.023

Table 3.17: Re-generating the erased positions using argmax in Tokenwise rejection sampling with backtracking (Algorithm 1) slightly improves the accuracy-diversity tradeoff (Section 3.2.2.1) in our test case generation task. We fixed nucleus sampling (Holtzman et al., 2020) top_p = 0.95, backtrack quota $Q = 4$, and backtrack stride $B = 4$ (the best settings in Table 3.13). The row "ablation: no argmax" refers to removing lines 9-12 in Algorithm 1. The column *err threshold* denotes the cutoff below which the error predictor output is interpreted as a rejection (Section 3.2.2.6). The experiment was repeated 5 times, and we report the standard errors.

3.3 Theoretically provable algorithm: value-guided sampling with stochastic backtracking

Section 3.2 introduced a natural heuristic algorithm based on *backtracking* (Botta et al., 2025; Yang et al., 2025; von Rütte et al., 2025)—i.e. occasionally "erasing" generated tokens. Despite showing promising empirical results, it has not been implemented in a mathematically justified manner, and its benefits have not been rigorously quantified. The theory in Section 3.1 highlights the benefits (both information-theoretical and computational) of process verifiers, but does not provide theoretical guaranty when the process verifier is imperfect.

In this section (based on Rohatgi et al. (2025)), we introduce a concrete model for imperfect process verifiers, in which we show error amplification *can be algorithmically mitigated.* To do so, we connect verifier-assisted language generation with the classical algorithmic toolkit for sampling—specifically, a Markov chain Monte Carlo (MCMC) technique from theoretical computer science that leverages *approximate counting* oracles to do approximate sampling (Sinclair & Jerrum, 1989). This machinery lets us implement the empirical heuristic of *backtracking* in a mathematically justified manner, and to establish rigorous guarantees for guiding generation with an imperfect process verifier. Broadly, we believe that perspectives from classical theory on design and analysis of Markov chains may bear additional fruits for language model reasoning , and we hope our paper will stimulate further work to connect these areas.

More concretely, the algorithm which we develop, `VGB`, is based on a principled *stochastic backtracking* strategy that generalizes the seminal *Sinclair-Jerrum random walk* (Sinclair & Jerrum, 1989).

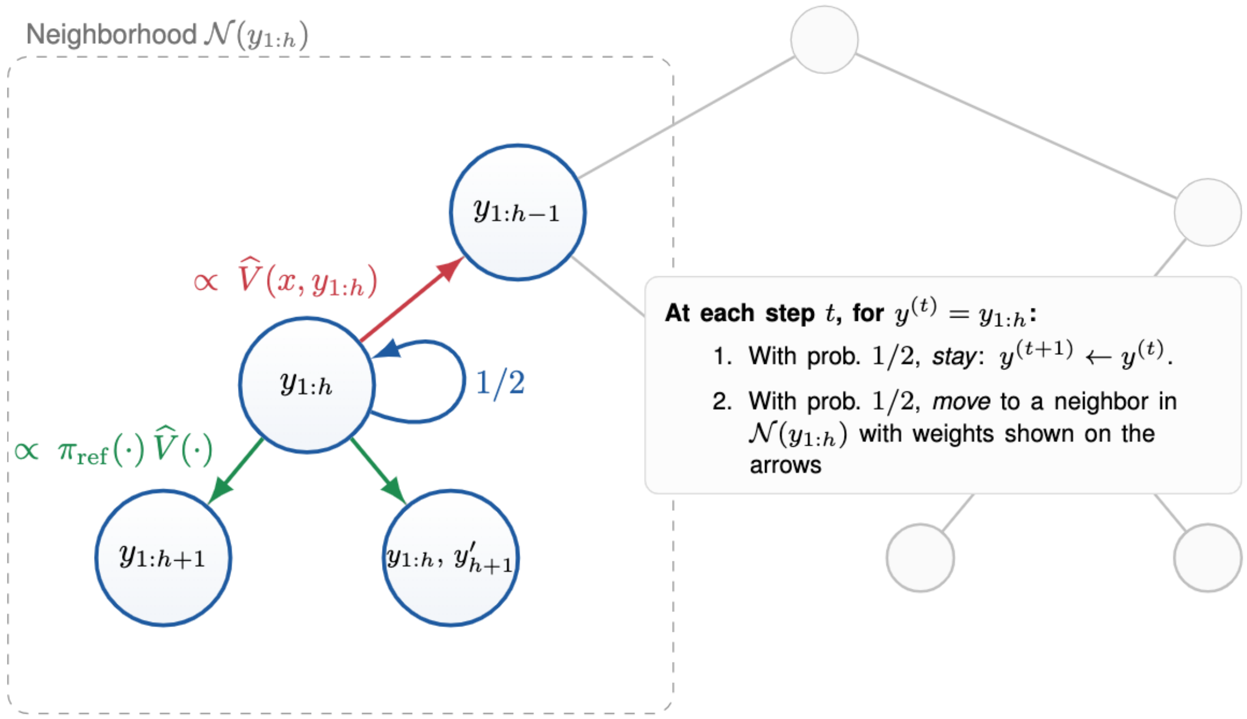
3.3.1 `VGB`: the Value-Guided Sampling with Stochastic Backtracking algorithm

Algorithm 2 VGB: Value-Guided Sampling with Stochastic Backtracking

```
Input: Base model $\pi_{\mathrm{ref}}$; appx. value function $\widehat{V}$, prompt $x \in \mathcal{X}$, horizon $H \in \mathbb{N}$, step count $T \in \mathbb{N}$.
If outcome-level reward/tilt is available, set $\widehat{V}(x, y_{1:H}) := \tau(x, y_{1:H})$.
Initialize $y^{(0)} := \emptyset$.
for $0 \le t < T$ do
    Set $h := |y^{(t)}|$, and define $p^{(t)}$ as the distribution over neighborhood $\mathcal{N}(y^{(t)}_{1:h})$ of $y^{(t)}_{1:h}$ where
        $p^{(t)}(y^{(t)}_{1:h-1}) \propto \begin{cases} \widehat{V}(x, y^{(t)}_{1:h}) & \text{if } h > 0, \\ 0 & \text{if } h = 0, \end{cases}$
        and for each $y_{h+1} \in \mathcal{A}$,
        $p^{(t)}(y^{(t)}_{1:h}, y_{h+1}) \propto \begin{cases} \pi_{\mathrm{ref}}(y_{h+1} \mid x, y^{(t)}_{1:h}) \widehat{V}(x, y^{(t)}_{1:h}, y_{h+1}) & \text{if } h < H, \\ 0 & \text{if } h = H. \end{cases}$
    With probability 1/2, set $y^{(t+1)} := y^{(t)}$, else sample $y^{(t+1)} \sim p^{(t)}$.
end for
return $y^{(T)}$ or $y^{(i)}$ for $i \sim \mathrm{Unif}([T])$.
```

Our main algorithm, `VGB`, is displayed in algorithm 2 and illustrated in fig. 3.5. To explain the algorithm, we view guided sampling algorithms as implicitly traversing the exponentially-large tree $\mathcal{T}$ with node set $\bigcup_{h=0}^{H} \mathcal{A}^h$, where $y_{1:h-1}$ is the parent of $y_{1:h}$ (for any $y_i \in \mathcal{A}$). Let $\mathcal{N}(y_{1:h})$ denote the neighborhood of $y_{1:h}$ in $\mathcal{T}$, which contains its parent $y_{1:h-1}$ and all its children $\{y_{1:h+1}\}_{y_{h+1} \in \mathcal{A}}$.

`VGB` proceeds as follows. At each step t, given the current node $y^{(t)}_{1:h}$, we first define a probability distribution $p^{(t)}$ over the neighborhood $\mathcal{N}(y^{(t)}_{1:h})$. The probability of selecting a child node $(y^{(t)}_{1:h}, y_{h+1})$ is proportional


Neighborhood $\mathcal{N}(y_{1:h})$
$y_{1:h-1}$
$\propto \widehat{V}(x, y_{1:h})$
$y_{1:h}$
$1/2$
$\propto \pi_{\text{ref}}(\cdot)\,\widehat{V}(\cdot)$
$y_{1:h+1}$
$y_{1:h}, y'_{h+1}$

At each step t, for $y^{(t)} = y_{1:h}$:

1. With prob. $1/2$, *stay*: $y^{(t+1)} \leftarrow y^{(t)}$.
2. With prob. $1/2$, *move* to a neighbor in $\mathcal{N}(y_{1:h})$ with weights shown on the arrows

Figure 3.5: Illustration of execution of `VGB` at each step t.

to $\pi_{\texttt{ref}}(y_{h+1} \mid x, y^{(t)}_{1:h})\hat{V}(x, y^{(t)}_{1:h}, y_{h+1})$, while the probability of selecting the parent $y^{(t)}_{1:h-1}$ is proportional to $\hat{V}(x, y^{(t)}_{1:h})$. The random walk proceeds by sampling a new node $y^{(t+1)}$ from $p^{(t)}$ with probability $1/2$, and staying at the current node otherwise (setting $y^{(t+1)} \leftarrow y^{(t)}$).[12] After T steps, in the version of `VGB` that we analyze first, it returns the final node of the random walk. In experiments, we return the first leaf node the algorithm reaches (for efficiency), but the version of the algorithm with theoretical guaranty depends on returning a uniformly chosen node from the random walk's path. [13]

`VGB` as a generalization of the Sinclair-Jerrum walk `VGB` can be interpreted as a generalization of the Sinclair-Jerrum walk (Sinclair & Jerrum, 1989), which corresponds to the special case where $\pi_{\texttt{ref}}$ is uniform and τ is binary-valued.[14] This walk was originally used to show that approximate *sampling* reduces to approximate *counting* in self-reducible problems such as SAT, without incurring the error amplification of prior reductions (Jerrum et al., 1986). Value functions are precisely the generalization of counting oracles to our setting.

Theoretically, under some assumptions, `VGB` enjoys provable guarantees on its sampling quality and efficiency. More concretely, `VGB` inherits three key properties from the Sinclair-Jerrum walk: (1) the walk rapidly mixes to a stationary distribution $\tilde{\pi}$; (2) $\tilde{\pi}$ puts $\Omega(1/H)$ mass on the leaves of $\mathcal{T}$; and (3) as long as exact outcome-level rewards are used (i.e. $\hat{V}(x, y_{1:H}) = \tau(x, y_{1:H})$), $\tilde{\pi}$ is proportional to π^* at the leaves.

Empirically, `VGB` achieves favorable quality-efficiency trade-off on various synthetic and real constrained generation tasks, including but not limited to Dyck grammar and Python code generation, which we introduced in Section 3.2.

[12] This technique (staying at the current node with probability $1/2$) is called *laziness*, and is needed to ensure that the random walk has a stationary distribution. See e.g. Levin et al. (2009) for background on Markov chains.

[13] The chosen node may not be a leaf, in which case we re-run the algorithm; this can occur at most $\tilde{O}(H)$ times.

[14] See Bakshi et al. (2024) for a related generalization, applied to quantum Gibbs state preparation.

For more details on the theoretical guarantees and the empirical results, please refer to our paper (Rohatgi et al., 2025).

3.4 Discussions

3.4.1 Is query efficiency a reasonable notion of efficiency?

There are many reasonable efficiency metrics, and they do not always positively correlate with each other (Dehghani et al., 2021).

Our paper focuses on *query complexity* (measured by the number of tokens generated by the language model to satisfactorily complete the task [15]), and we do not claim that the same conclusions apply when we switch out query complexity for other metrics of efficiency, such as wall-clock time.

We think query complexity is one (but not necessarily the only, or the most) important aspect of efficiency due to the following considerations:

- Many existing large language model (LLM) providers charge service fees to the users according to the number of tokens generated by the language model for the user, i.e. query complexity.

- In the *single sequence generation* setting, controlling all other conditions to be held the same, query complexity positively correlates with the size of computation (the number of decoder forward passes) and wall-clock time.

- In the *batched generation* setting, admittedly, the wall-clock time does not necessarily scale linearly with query complexity [16] , meaning that the naive best-of-n rejection sampling is not as slow as query complexity would indicate (if the LLM has sufficient bandwidth for it). However, in many realistic LLM inference settings, the LLM receives a large number of query requests per second, so there is no additional idle availability [17] for duplicating each sequence generation request by n.

Although, as mentioned above, query complexity is partially indicative of a few practically important efficiency metrics (e.g. monetary cost or wall-clock time), there are aspects of these metrics that are not tracked by query complexity. For example, different types of *hardware* and *cache* may have different efficiency best practices. In particular, on GPUs and TPUs, algorithms that better exploit *parallelization* or *tensorized computation* tend to be more efficient. Therefore, an important direction for future work is to design and analyze *hardware-aware algorithms* that incorporate these important aspects of the inference setup.

[15] This definition is natural since generating one token involves one forward pass of the (decoder-only autoregressive) language model, i.e. one query.

[16] For example, the wall-clock time of generating n candidate responses (with batch size n) might be less than n multiplying the wall-clock time of generating 1 candidate response.

[17] Unless more GPUs/TPUs are allocated to serve this LLM.

3.5 Proof intuitions

3.5.1 On the hardness of the knapsack problem

The hardness of the knapsack problem has been the subject of extensive study. Specifically, the decision version of this problem has found applications in the context of secure cryptosystems Odlyzko (1998). Under no assumptions on the input structure, the best known algorithm is based on dynamic programming Kellerer et al. (2004) and runs in pseudo-polynomial time. This algorithm is also used to obtain an FPTAS and its runtime is effectively polynomial if one further assumes that the weights are polynomially bounded in D. More exact or approximate algorithms achieve polynomial runtime, under specific input structures. Specifically, when the weights form a superincreasing sequence, that is,

$$X_i \geq \sum_{j=1}^{i-1} X_j \quad \forall i \in [2, D] \cap \mathbb{Z},$$

a greedy algorithm solves the knapsack decision problem Odlyzko (1998) in linear time. On the other hand, when the density of the knapsack

$$\frac{D}{\log_2(\max_i \{X_i\}_{i=1}^d)}$$

is small enough, knapsack is approximately solved in polynomial time by lattice reduction algorithms Plantard et al. (2013). Our argument considers the most general setting, in which no assumptions are made on the structure of the inputs $\{X_i\}_{i=1}^t$, c and the decision problem is NP-complete Karp (1972).

3.5.2 Proof intuition for Theorem 3.1.1

Remark 3.5.1. *To help readers parse our proof above, we provide its informal intuition. The oracle $\mathcal{O}_{\hat{s}}$ can be thought of as hiding a secret message $\hat{s}$ which is a binary sequence of length $D-1$. Because of our construction in Equation (3.1), by querying the oracle with any string $s \neq \hat{s}$, the output of the oracle will not reveal any information about $\hat{s}$. Therefore, to know anything about $\hat{s}$, the query s needs to exactly match $\hat{s}$. For any deterministic order of searching for $\hat{s}$ over $\{0,1\}^{D-1}$, the worst case is always that $\hat{s}$ is the last item in the search order, causing runtime 2^{D-1}.*

Moreover, Theorem 3.1.1 generally applies to any algorithm $\mathcal{A}$. In particular, $\mathcal{A}$ is even allowed to never *query the generator oracle at all. Intuitively, one candidate counter-example of our theory would be a simple algorithm $\mathcal{A}$ which always outputs 0 at all positions (as this will obviously satisfy the counstraint $A := \{s \in \Sigma^D : \sum_{i=1}^D s_i \mod 2 = 0\}$). However, recall that Definition 3.1.2 requires that the output must have nonzero probability under the generator oracle $\mathcal{O}_{\hat{s}}$. Note that $s_i = 0 \ \forall i \in [D]$ will not have nonzero probability under $\mathcal{O}_{\hat{s}}$, thus violating the constraints (unless $\hat{s}_i = 0 \ \forall i \in [D-1]$). The intuitive reason why the above counter-example does not work is that, it is necessary to use the oracle $\mathcal{O}_{\hat{s}}$ to know the $\hat{s}$ that it hides. Therefore, any algorithm $\mathcal{A}$ is governed by the above-mentioned lower bound of searching for $\hat{s}$ by querying $\mathcal{O}_{\hat{s}}$.*

3.6 Related work

Inference-time scaling for language models Practical language generation tasks typically impose various task-specific constraints in addition to the general grammatical rules of language. One effective way to improve the chance of satisfying such constraints is to increase the inference-time compute through search and/or rejection sampling. There has been a long history of prior works that employ inference-time scaling in the language generation context, dating as far back as beam search (Lowerre & Reddy, 1976; Hayes-Roth et al., 1976; Ow & Morton, 1988; Jurafsky & Martin, 2000; Graves, 2012). Much more recently, as researchers develop the techniques for language models to follow instructions (see the survey by Zhang et al. (2023a) and references therein), more creative designs for inference-time scaling algorithms have become viable (Wang et al., 2022; Yao et al., 2023; Zhang et al., 2023b; Zhou et al., 2023; Choi et al., 2023; Liu et al., 2024; Xie et al., 2024; Snell et al., 2024; Zhao et al., 2024), and see Wu et al. (2024) for a recent survey on cost-performance tradeoffs of these approaches. Compared to these approaches in the literature, our Tokenwise rejection sampling with backtracking (Algorithm 1) shares some features with lookahead search (Snell et al., 2024) (specifically, the rejection decision at the current position is based on the verifier decision at some future position). However, two main differences are: (1) Tokenwise rejection sampling with backtracking (Algorithm 1) does not use a beam (i.e. does not need to generate multiple candidates, thus reducing query complexity), and (2) Algorithm 1 uses a different sampling approach (namely argmax) for the backtracked positions (we verify in Section 3.2.3 that in some settings this significantly improves the accuracy). It is a natural future research direction to design inference algorithms that combine the advantages of the two.

Incorporating a process reward model to assist language generation Among the vast design space for inference-time scaling, process reward modeling has been proven to be an important component common to many LLM systems (Polu & Sutskever, 2020; Uesato et al., 2022; Ma et al., 2023; Lightman et al., 2023; Wang et al., 2024; Setlur et al., 2024). The process verifier which we study (Definition 3.1.3) is a special case of such process reward model if we restrict the output to be binary. However, there are still challenging open problems around process reward modeling, such as how to properly define the “blocks” (Guo et al., 2025) (see also our definitions in the “Block verifier” part of Section 3.2.2.6). Towards bringing more clarity to these open questions, our work develops a theoretical framework for reasoning about the query complexity of process verifiers. Moreover, our experiments suggest the potentials of a lightweight process verifier in improving the query complexity, accuracy, and diversity of constrained generation. In particular, our theory and experiments suggest (1) the “blocks” do not necessarily have to be carefully designed — setting each token as a block might potentially suffice, at least in some more structured domains such as codes; (2) *backtracking* (Algorithm 1, Section 3.2) is a robustly effective strategy that should be applied in combination with process verifiers. A possible extension of the type of verifier we study (Definition 3.1.3) is: instead of outputting binary acceptance / rejection decisions, the verifier could return a probability of accepting each prefix Yang & Klein (2021). However, some tasks may require that the output distribution should match some target distribution, and it may be challenging to ensure that the acceptance probability is well-calibrated in order to satisfy this requirement. Relevant to this goal, Foster et al. (2025) proves some theoretical guarantee for the end-to-end learning of process verifiers in some specialized settings.

Controlled synthetic data distribution as a sandbox for studying language models Our Dyck grammar distribution most closely follows Wen et al. (2023) (though we switched to a fixed-sequence-length setting, and used unbalanced bracket type probability, instead of length extrapolation, to define the criteria for a prompt to be *out-of-distribution*). Dyck grammar was also used in other prior works (Hewitt et al., 2020; Ebrahimi et al., 2020; Yao et al., 2021; Liu et al., 2023c;a) to study language models. Dyck grammar can be seen as a special case of the task (specifically context-free grammar) considered in SynCode (Ugare et al., 2024). Other synthetic data distributions have been used to study various aspects of language models in prior works, including representational capability (Bhattamishra et al., 2020a; Li & Risteski, 2021; Zhang et al., 2022a; Zhao et al., 2023), statistical sample complexity (Edelman et al., 2022), optimization process (Lu et al., 2021; Jelassi et al., 2022; Li et al., 2023; Bietti et al., 2023), sampling (Li et al., 2024b), and architectural limitations (Liu et al., 2023b), and see references cited therein.

Other related works include Huang et al. (2025a;b), who provide algorithms and theoretical guarantees for imperfect *outcome-level* verifiers, and Balcan et al. (2025), who study the sample complexity of learning outcome-level verifiers for chain-of-thought reasoning.

3.7 Conclusion

We introduce a new theoretical framework for elucidating the design space of verifiers and correspondingly a simple family of rejection-sampling-based inference algorithms. In particular, our theory proves the computational benefits of incorporating a *process verifier*, measured by the *query complexity* of calling the generator. On the other hand, our theory also reveals the subtleties: straightforwardly applying a process verifier in a Tokenwise rejection sampling algorithm may unintentionally re-weigh the distribution among sequences that satisfy the constraints, which could be undesirable for settings that require a strong notion of distributional *calibration*. Towards that goal, our theoretically principled algorithm, value-guided sampling with stochastic backtracking, incorporates classic results from the approximate counting and sampling literature and enjoys provable guarantees on the distributional calibration and computational efficiency.

Empirically, through fine-grained experiments on both synthetic and realistic data, we show that the Tokenwise rejection sampling algorithm, when combined with *backtracking*, is a robustly effective recipe for reducing query complexity, improving accuracy, and maintaining diversity. Moreover, value-guided sampling with stochastic backtracking mitigates error amplification as sequence length grows.

For future works, we hope the theoretical framework and empirical observations can inspire systematic characterization of the strengths and weaknesses of the diverse set of rejection-sampling-based inference-time algorithms. Concrete open problems at the intersection of theory and experiments include (1) improving the efficiency of backtracking-based inference-time algorithms (e.g. improving parallelization), and (2) improving the robustness of verifier-assisted language generation algorithms which require weaker assumptions on the verifier.

Chapter 4

Co-designing inference and training procedures for parallel-efficient language models

Chapter 2 and Chapter 3 considered training dynamics and inference algorithms, respectively. In both cases, our analysis is based on identifying key structures in the data distribution (such as topic or grammar), and studying how they are learned by the neural network (during training) or leveraged (during inference). These results intuitively suggest that the structural properties of the data distribution are the centerpiece which connects some aspects of training and inference. In this chapter (based on Li et al. (2024b)), we verify this intuition, by explicitly proving such connection. Specifically, we prove that the statistical efficiency of training can be bounded by the computational efficiency of inference. Moreover, these notions of efficiency are governed by a property of the data distribution. These results suggest that data, training, and inference are fundamentally connected, and studying their interactions is an important direction of research.

Autoregressive language models are the currently dominant paradigm for text generation, but they have some fundamental limitations that cannot be remedied by scale—for example inherently sequential and unidirectional generation. While alternate classes of models have been explored, we have limited mathematical understanding of their fundamental power and limitations. In this paper we focus on *Generative Masked Language Models (GMLMs)*, a non-autoregressive paradigm in which we train a model to fit conditional probabilities of the data distribution via masking, which are subsequently used as inputs to a Markov Chain to draw samples from the model. These models empirically strike a promising speed-quality trade-off as each step can be typically parallelized by decoding the entire sequence in parallel. We develop a mathematical framework for analyzing and improving such models which sheds light on questions of sample complexity and inference speed and quality. Empirically, we adapt the T5 model for iteratively-refined parallel decoding, achieving 2-3x speedup in machine translation with minimal sacrifice in quality compared with autoregressive models. We run careful ablation experiments to give recommendations on key design choices, and make fine-grained observations on the common error modes in connection with our theory. Our mathematical analyses and empirical observations characterize both potentials and limitations of this approach, and can be applied to future works on improving understanding and performance of GMLMs. [1]

The current dominant approach to language modeling is *autoregressive* (AR): to generate a sequence of tokens, the language model starts by predicting the leftmost token, and then proceeds from left to right, each step predicting the next token based on everything on its left (Raffel et al., 2020; Brown et al., 2020; Touvron et al., 2023). AR models are not without limitations:

1. *Lack of parallelism:* To generate a sequence of N tokens, AR language models need N sequential decoding steps. Each step consists of a forward pass of the decoder component. When N is large, N

[1] Our codes are released at `https://github.com/google-research/google-research/tree/master/padir`

sequential decoding steps incur high latency.

2. *Quality:* When predicting each token, the model cannot access its right hand side context, and has no natural way to revise earlier predictions on the left. This intuitive limitation was more formally explored in prior theoretical works (Li & Risteski, 2021; Lin et al., 2021; Bachmann & Nagarajan, 2024).

One promising alternative is based on *Generative Masked Language Models (GMLMs)*. They are trained to fit conditional probabilities for parts of the sequence (by applying a mask), conditioned on the rest. To produce samples, these conditionals are used as oracles for running Markov Chain, e.g. a Gibbs sampler (Wang & Cho, 2019; Goyal et al., 2022). Alternatively, we can think of these steps as an iterative refinement process, typically starting with pure noise (i.e. all tokens are masked or randomized). One can even fit conditional probabilities for noised versions of the input distribution, and use them as inputs to a denoiser to get certain types of discrete diffusion models (Austin et al., 2021). In GMLMs, typically one step of the Markov Chain is operationalized by a Transformer that generates the sequence in parallel (i.e. *parallel decoding* (Ghazvininejad et al., 2019; Gu & Kong, 2021; Savinov et al., 2022)). Hence, if the total number of steps is small, the latency is low.

However, none of these approaches robustly surpass autoregressive models in both speed and quality for a wider range of language generation tasks beyond machine translation. Thus, the following questions naturally arise:

(Q1) GMLMs are trained to learn conditional probabilities. When does it also imply learning the *joint* probability?

(Q2) What properties of the data distribution and training/inference algorithm govern the quality of the learned model and its generated samples?

(Q3) What are the best practices for training GMLMs, and can we use theory to elucidate the design space of losses, training and inference procedures?

Our contributions. Towards answering the questions above, we introduce a *theoretical framework* to characterize the potentials and limitations of GMLMs, for both training and inference. Precisely,

- The **asymptotic sample complexity** for estimating the parameters of a distribution via a broad class of masked-prediction losses can be related to the mixing time of a corresponding Markov Chain that can be used to sample from the distribution (Section 4.1.2). In particular, we prove that training with larger masks always improves statistical efficiency (Theorem 4.1.1).
- We show **finite-sample bounds** that relate how well the *conditional* distributions of the data distribution are learned, to how well the *joint* distribution is learned (Section 4.1.3) if we have some capacity control over the distribution class being learned (e.g. covering number bounds).
- **Transformers** are only able to represent decoding steps that factorize over the coordinates—preventing them from **efficiently sampling** even simple distributions with strong correlations between the coordinates (Section 4.1.4).

We accompany these theoretical findings with an extensive set of empirical investigations detailing important components and common error modes. Precisely:

- Our experiments (Section 4.2) suggest the **empirically critical components** include large masking ratio (c.f. theory in Section 4.1.2), custom vocabulary, distillation from AR models, and architecture improvements like positional attention. [2]

[2] The benefit of distillation was verified in Kim & Rush (2016); Gu et al. (2018); Zhou et al. (2020); Gu & Kong (2021). Positional attention was tested in Gu et al. (2018); Kreutzer et al. (2020).

- GMLMs with parallel-decoding work well on **machine translation**: in fact, even *one single* forward pass can often produce reasonable translations. This aligns with our theoretical framework, as machine translation tasks typically involve lower-entropy and less multi-modal outputs, compared to other language generation tasks.

- Common **error modes** ("stuttering") suggest limitations for parallel-decoding GMLMs for modeling strong dependencies (c.f. theory in Section 4.1.4), which we empirically quantify (Section 4.2.4).

Jointly, our theoretical and empirical findings suggest synergistically designing better Markov Chains that mix fast in the presence of strong correlations in the target, and corresponding losses that inherit good statistical behavior.

4.1 Theoretical framework

We develop a mathematical framework for reasoning about the core ingredients for successfully training and using GMLMs: the *statistical complexity* to learn the model, and the *speed of inference*. We show that these two are surprisingly closely related: namely, we understand both the asymptotic and finite-sample statistical complexity through *functional inequalities* (e.g. Poincaré, approximate tensorization of entropy) corresponding to the Markov Chains we would use at inference time—which in turn characterize the mixing time of these chains. This picture closely mirrors an emerging picture in the continuous case for score-based (diffusion) models (Koehler et al., 2023; Qin & Risteski, 2023)—though with somewhat different proof techniques.

4.1.1 Setup and notation

The most classical way of fitting distributions from data is maximum likelihood: that is, finding the choice of parameters that maximize the likelihood of the training data. There are well-understood statistical reasons to do so: in the asymptotic sense (as the number of sample grows), maximum likelihood is the most sample-efficient way to estimate the distribution (Hájek, 1972). However, many families of distributions are such that optimizing maximum likelihood is computationally challenging. Thus, many alternate strategies and losses to fit the parameters have been developed.

For continuous distributions, a common choice of loss is the *score matching* loss, where instead of fitting the likelihood, we fit the gradient with respect to the input of the log-pdf, namely $\nabla_x \log p_\theta(x)$. In certain cases, this can provable computational benefits over maximum likelihood (Pabbaraju et al., 2023). For discrete distributions, we cannot take gradients with respect to the input: though a closely related strategy is available — trying to match the conditionals of subsets of variables. (This can be thought of as "flipping" the coordinates in the subsets, while keeping the remaining coordinates fixed.) Operationalizing this as a loss gives us the *pseudolikelihood* loss (Besag, 1975).

Variants of this strategy have been used in classical results for learning Ising models (Ravikumar et al., 2010; Vuffray et al., 2016). More recently, this strategy has been used in conjuction with neural models to both learn useful features in the guise of masked language modeling (MLM) (Devlin et al., 2019), which can be also used to produce a generative model (Wang & Cho, 2019; Goyal et al., 2022). The latter is done by using the learned conditionals inside a Gibbs sampler. However, when the conditionals are not *consistent*, i.e. there is not a joint distribution that satisfies these conditionals, Gibbs sampler may amplify errors. In general, mathematical understanding about sampling from masked language models is still lagging substantially behind.

Setup and notation: Let Ω be a finite discrete set. Let $p_{\mathcal{X}}$ denote a distribution over a sequence of N variables $X = (X_1, X_2, \cdots, X_N) \in \Omega^N =: \mathcal{X}$. [3] Furthermore, for $K \subset [N]$, let X_K denote the subsequence $(X_i \,|\, i \in K)$, and X_{-K} denote the subsequence $(X_i \,|\, i \notin K)$.

[3] In language models, Ω is the set of tokens in the vocabulary.

We consider learning parameters θ parametrizing some distribution p_θ over $\mathcal{X}$, for $\theta \in \Theta$. The classical way of fitting θ is to maximize the likelihood of the training data:

Definition 4.1.1 (MLE, Van der Vaart (2000))**.** *Given i.i.d. samples $x_1, \ldots, x_n \sim p_{\theta^*}$, the maximum likelihood estimator is $\hat{\theta}_{MLE} = \arg\max_{\theta \in \Theta} \hat{\mathbb{E}}\left[\log p_\theta(X)\right]$, where $\hat{\mathbb{E}}$ denotes the expectation over the samples. As $n \to \infty$ and under suitable regularity conditions, we have $\sqrt{n}\left(\hat{\theta}_{MLE} - \theta^*\right) \to N\left(0, \Gamma_{MLE}\right)$, where $\Gamma_{MLE} := \mathcal{I}^{-1}$, $\mathcal{I} := \mathrm{Cov}_{X \sim p_{\theta^*}}\left(\nabla_\theta p_\theta(X)\right)_{|\theta=\theta^*}$ is the Fisher information matrix.*

A classical result due to Hájek-Le Cam (for modern exposition see Van der Vaart (2000)) is that maximum likelihood is the asymptotically most sample-efficient estimator among all "sufficiently regular" estimators (Section 8.5 in Van der Vaart (2000)) — so we will treat it as the "gold standard" against which we will compare other estimators. The class of estimators we will be focusing most is the a broad generalization of the *pseudo-likelihood estimator* (Besag, 1975).

Definition 4.1.2 (Weighted pseudolikelihood)**.** *Consider a partition of $[N]$, namely a collection of sets $\mathcal{K} := \{K_1, \ldots, K_{|\mathcal{K}|}\}$ such that $\bigcup_i K_i = [N]$, and a distribution $p_\mathcal{K} : \mathcal{K} \to \mathbb{R}^+$.*

Given n i.i.d samples of sequences and masks: $\{\left(X^{(i)}, K^{(i)}\right) | X^{(i)} \sim p_\mathcal{X}, K^{(i)} \sim p_\mathcal{K}\}_{i \in [n]}$, the weighted maximum pseudolikelihood estimator (MPLE) *is $\hat{\theta}_{PL} := \arg\min_\theta \sum_{i=1}^n - \log p_\theta(X^{(i)}_{K^{(i)}} | X^{(i)}_{-K^{(i)}})$. The population loss is* [4] *$L_{PL}(\theta) := \mathbb{E}_{X \sim p_\mathcal{X}, K \sim p_\mathcal{K}}\left[- \log p_\theta(X_K | X_{-K})\right]$.*

As a special case, if $\mathcal{K}$ contains all subsets of a certain size k and $p_\mathcal{K}$ is uniform over $\mathcal{K}$, we get the classical k-pseudolikelihood estimator:

Definition 4.1.3 (k-pseudolikelihood (Huang & Ogata, 2002))**.** *Same as Definition 4.1.2 except that $\mathcal{K} := \{K \subseteq [N] \mid |K| = k\}$, $p_\mathcal{K} = Unif(\mathcal{K})$.*

Remark 4.1.1. *The distribution of X and K in the above loss is independent. In Section 4.1.2.3 we will show that our results readily generalize to losses in which the distribution of the masks K can depend on the current X. We present the independent case first for ease of presentation.*

Informally, we predict the variables in positions $K \in \mathcal{K}$, conditioned on the remaining variables. The benefit is that parametrizing conditionals over smaller subsets K is often computationally cheaper. For instance, if $p_\theta(x)$ is an undirected graphical model, i.e. $p_\theta(x) \propto \exp(\sum_C \phi_{C,\theta}(x_C))$, where the sum is over all maximal cliques C of the graph describing the distribution, the conditional distribution of K only depends on its Markov blanket, which can be very small for sparse graphs and small sets K. Thus, computing the partition function corresponding to $p(x_K | x_{-K})$ takes time exponential in this Markov blanket. By contrast, computing the likelihood requires calculating the partition function of $p_\theta(x)$, which takes time exponential in the dimension of X. In fact, for Ising models, the corresponding loss is even *convex* [5]. A similar tradeoff exists for masked language models: fitting the conditionals for larger masks would likely require a larger model, thus would be computationally more expensive.

4.1.2 Asymptotic sample efficiency via functional inequalities

In this section, we will provide a framework for bounding the asymptotic sample complexity of learning the parameters θ of a discrete probability distribution by minimizing a loss in a broad family of "masked prediction" objectives. We will measure the quality of an estimator in terms of *parameter recovery.* To make this formal, we first recall that under mild technical conditions, the estimator will be asymptotically normal:

Lemma 4.1.1 (Asymptotic normality (Van der Vaart, 2000))**.** *Consider the weighted MPLE objective in Definition 4.1.2, and let $\theta^* \in \arg\min_\theta L_{PL}(\theta)$. Under mild regularity conditions (Lemma 4.3.9 in Section 4.3.10), as $n \to \infty$, $\sqrt{n}(\hat{\theta}_{PL} - \theta^*) \xrightarrow{d} \mathcal{N}(0, (\nabla^2_\theta L_{PL}(\theta^*))^{-1} \mathrm{Cov}(\nabla_\theta l_{PL}(\theta^*))(\nabla^2_\theta L_{PL}(\theta^*))^{-1})$*

[4] This is equivalent to minimizing the KL divergence of the ground-truth conditional distribution $p(X_K | X_{-K})$ from the predicted conditional distribution $p_\theta(X_K | X_{-K})$: $\mathbb{E}_{X \sim p_\mathcal{X}, K \sim p_\mathcal{K}}\left[D_{\mathrm{KL}}\left(p(\cdot | X_{-K}), p_\theta(\cdot | X_{-K})\right)\right]$

[5] This fact is well known, but for completeness included in Section 4.3.11

If we know $\sqrt{n}(\hat{\theta}_{PL} - \theta^*) \xrightarrow{d} \mathcal{N}(0, \Gamma_{PL})$, we can extract bounds on the expected ℓ_2^2 distance between $\hat{\theta}_n$ and θ^*. Namely, from Markov's inequality, (see e.g., Remark 4 in Koehler et al. (2023)), for sufficiently large n, with probability at least 0.99 it holds that $\|\hat{\theta}_{PL} - \theta^*\|_2^2 \le \frac{\mathrm{Tr}(\Gamma_{PL})}{n}$.

4.1.2.1 Masking more is (statistically) better

As a first application of our framework, we prove that increasing the number of variables k we predict in k-pseudolikelihood (Definition 4.1.3) strictly improves the statistical efficiency of the resulting estimator. Note, for larger k, we expect the computational cost to optimize the corresponding loss to be larger, and when $k = N$ we just get max likelihood. Thus, this naturally formalizes a *computational/statistical tradeoff* in choosing k.

Assumption 4.1.1 (Finite gradient and Hessian)**.** *$\forall \theta \in \Theta, \forall x \in \mathcal{X}, K \subset [N]$, the norms of the gradient $\|\nabla_\theta \log p_\theta(x_K|x_{-K})\|_2$ and the Hessian $\|\nabla_\theta^2 \log p_\theta(x_K|x_{-K})\|_F$ exist and are finite .*

Assumption 4.1.2 (Realizability)**.** *The data distribution $p_{\mathcal{X}}$ satisfies: $\exists \theta^* \in \Theta$ such that $p_{\theta^*} = p_{\mathcal{X}}$.*

Theorem 4.1.1 (Masking more is (statistically) better)**.** *Let Assumption 4.1.1 and Assumption 4.1.2 be satisfied, and let Γ_{PL}^k denote the asymptotic variance of the k-MPLE estimator (Definition 4.1.3). Then, we have:*[6] $\Gamma_{PL}^{k+1} \preceq \Gamma_{PL}^k$.

Remark 4.1.2. *By monotonicity of trace, Thm 4.1.1 implies $\mathrm{Tr}(\Gamma_{PL}^{k+1}) \le \mathrm{Tr}(\Gamma_{PL}^k)$. By the remarks after Lemma 4.1.1, larger k implies a better asymptotic l_2 bound for learning θ since $\mathbb{E}_{X_{1:n}, K_{1:n}}\left[\|\hat{\theta}_{PL}^k - \theta\|_2^2\right] \to \frac{\mathrm{Tr}(\Gamma_{PL}^k)}{n}$.*

The main lemma needed for Theorem 4.1.1 is that the two matrices in the asymptotic covariance of MPLE, $\nabla_\theta^2 L_{PL}(\theta^*)$ and $\mathrm{Cov}_{X \sim p_{\mathcal{X}}, K \sim p_{\mathcal{K}}}(-\nabla_\theta \log p_\theta(X_K|X_{-K}))_{|\theta=\theta^*}$ are actually equal. For MLE (namely, when $k = N$) this is well-known and called the *information matrix equality.* Proofs of Lemma 4.1.2 and Theorem 4.1.1 are in Section 4.3.1 and Section 4.3.2. We empirically verify Theorem 4.1.1 in Section 4.2.1.

Lemma 4.1.2 (Generalized information matrix equality)**.** *Under Assumption 4.1.1 and Assumption 4.1.2, the weighted pseudolikelihood loss (Definition 4.1.2) verifies: $\nabla_\theta^2 L_{PL}(\theta^*) = \mathrm{Cov}_{X \sim p_{\mathcal{X}}, K \sim p_{\mathcal{K}}}(-\nabla_\theta \log p_\theta(X_K|X_{-K}))_{|\theta=\theta^*}$.*

4.1.2.2 Statistical efficiency bounds via mixing time bounds

We could in general conceive of masking strategies where certain subsets of variables get masked with different probabilities. For instance, in language, nearby words will tend to be more correlated; grammatical constraints will dictate the parts-of-speech that can occur in different positions. We would then like to have theoretical guidance on what choices of masking distributions are better. Remarkably, it turns out that we can relate the statistical efficiency — in the sense of $\mathbb{E}\|\hat{\theta} - \theta^*\|^2$ for the resulting estimator $\hat{\theta}$ — and the mixing time of an appropriately chosen Markov Chain. In fact, this is the Markov Chain that would be typically chosen at inference time. Towards making this formal, we will need several preliminary concepts and results for Markov chains. Recall, a Markov chain on a state space Ω is described by a (row-stochastic) transition matrix P. Moreover, we can assign a natural bilinear form called the Dirichlet form:

Definition 4.1.4 (Dirichlet form)**.** *Let M be an ergodic, reversible Markov chain with transition matrix P on state space Ω. Let μ be its unique stationary distribution. $\forall f, g : \Omega \to \mathbb{R}$ the associated Dirichlet form is defined as:*

$$\mathcal{E}_P(f, g) := \langle f, (I - P)g \rangle_\mu = \frac{1}{2}\Sigma_{x,y\in\Omega}\mu(x)P(x,y)(f(x) - f(y))(g(x) - g(y))$$

Mixing time of the Markov chain can be bounded in the χ^2 sense by the gap between the 1st and 2nd eigenvalue of the Laplacian matrix $I - P$, expressed as Poincaré inequality:

[6] The notation $A \preceq B$ means $B - A$ is positive semidefinite.

Definition 4.1.5 (Poincaré inequality)**.** *We say that a Markov chain satisfies a* Poincaré inequality *with constant C if for all $f : \Omega \to \mathbb{R}$, we have $\mathcal{E}_P(f,f) \geq \frac{1}{C}\mathrm{Var}_\mu(f)$.*

The Poincaré inequality implies exponential ergodicity of the Markov chain in χ^2-divergence, precisely $\chi^2(p_t,\mu) \leq e^{-2t/C}\chi^2(p_0,\mu)$, where μ is the stationary distribution of the chain and p_t is the distribution after running the Markov process for time t, starting at p_0. We will be particularly interested in several generalizations of Gibbs sampling:

Definition 4.1.6 (Weighted block dynamics)**.** *Let $\mathcal{K} := \{K_1, \ldots, K_{|\mathcal{K}|}\}$ be a collection of sets (or blocks) such that $\bigcup_i K_i = [N]$. A block dynamics with blocks $\mathcal{K}$ is a Markov chain that picks a block K in each step according to some distribution $p_\mathcal{K} : \mathcal{K} \to \mathbb{R}^+$* [7] *and then updates the coordinates in K according to the conditional distribution $p_\mathcal{X}(X_K|X_{-K})$.*

The stationary distribution for the above Markov Chain is $p_\mathcal{X}$. Caputo & Parisi (2021) also derived the Dirichlet form (Definition 4.1.4) corresponding to this Markov chain:

$$\mathcal{E}(f,g) := \mathbb{E}_{K\sim p_\mathcal{K}}\mathbb{E}_{X_{-K}}\left[\mathrm{Cov}_{X_K|X_{-K}}(f,g)\right].$$

The crucial result we show is that the statistical efficiency of the weighted MPLE (Definition 4.1.2) as captured by the asymptotic variance can be related to the Poincaré constant of the corresponding weighted block dynamics (Definition 4.1.6). Proof of Theorem 4.1.2 is in Section 4.3.3.4.

Theorem 4.1.2 (Asymptotic variance under a Poincaré Inequality)**.** *Suppose the distribution p_{θ^*} satisfies a Poincaré inequality with constant C with respect to the weighted block dynamics. Then under Assumption 4.1.1 and Assumption 4.1.2 the asymptotic variance of the weighted MPLE can be bounded as: $\Gamma_{PL} \preceq C\mathcal{I}^{-1}$ where $\mathcal{I}$ is the Fisher Information matrix (Definition 4.1.1).*

4.1.2.3 Adaptive masking: masked positions depend on the sequence

The machinery we developed in Section 4.1.2.2 is in fact substantially more general — it applies to even "adaptive" masking losses in which the conditional distribution of the mask can depend on the current X (that is, for each X, there is a different conditional distribution $p_\mathcal{K}(K|X)$ which can be manually designed and is known to the model during training).

Definition 4.1.7 (Adaptively weighted pseudolikelihood)**.** *Given n i.i.d samples of sequences and masks: $\{\left(X^{(i)},K^{(i)}\right)|X^{(i)} \sim p_\mathcal{X}, K^{(i)} \sim p_\mathcal{K}(\cdot|X^{(i)})\}_{i\in[n]}$, the* weighted maximum pseudolikelihood estimator (MPLE) *is $\hat{\theta}_{PL} := \arg\min_\theta \sum_{i=1}^n -\log p_\theta(X_K|X_{-K},K)$, where*

$$p_\theta(X_K|X_{-K},K) := \frac{p_\theta(X)p_\mathcal{K}(K|X)}{\sum_{X'_K} p_\theta\left(X'_K, X_{-K}\right) p_\mathcal{K}(K|\left(X'_K,X_{-K}\right))} \tag{4.1}$$

The population loss is correspondingly:

$$L_{PL}(\theta) := \mathbb{E}_{X\sim p_\mathcal{X}}\mathbb{E}_{K\sim p_\mathcal{K}(\cdot|X)}\left[-\log p_\theta(X_K|X_{-K},K)\right].$$

Remark 4.1.3. *Note, the distribution $p_\mathcal{K}(\cdot|X)$ doesn't depend on θ, so $K^{(i)}$ can be generated readily by drawing samples from this distribution. Note also, the term $p_\theta(X_K|X_{-K},K)$ is expressible in terms of the joint distribution $p_\theta(X)$ and $p_\mathcal{K}(\cdot|X)$ and the expression in (4.1) can be interpreted as a conditional distribution in the joint distribution $p_{\theta,\mathcal{K}}(X,K) := p_\theta(X)p_\mathcal{K}(K|X)$. Finally, note that conditioning on the set K is subtle, but important — see Lemma 4.3.1 in Section 4.3.3.1.*

We can analogously generalize the sampling process to the following Markov chain in which K is sampled dependent on X:

[7] This is analogous to the training objective setting in Definition 4.1.2.

Definition 4.1.8 (Adaptive weighted block dynamics). *Let $\mathcal{K} := \{K_1, \ldots, K_{|\mathcal{K}|}\}$ be a collection of sets (or blocks) such that $\bigcup_i K_i = [N]$. A block dynamics with blocks $\mathcal{K}$ is a Markov chain that picks a block K in each step according to some distribution* [8] *$p_{\mathcal{K}}(\cdot \mid X)$, and then updates the coordinates in K according to the conditional distribution $p_{\mathcal{X}}(X_K|X_{-K}, K)$.*[9]

If we understand the domain of this Markov chain to be $\{(X, K) \mid X \in \mathcal{X}, K \in \mathcal{K}\}$, its stationary distribution is

$$p_{\mathcal{X},\mathcal{K}}(X, K) := p_{\mathcal{X}}(X)p_{\mathcal{K}}(K \mid X).$$

The Dirichlet form can also be explicitly written down (note, f and g are functions of both X and K):

Proposition 4.1.1 (Dirichlet form for adaptive weighted block dynamics). *The Dirichlet form corresponding to the weighted block dynamics (Definition 4.1.8) is:*

$$\mathcal{E}(f, g) = \mathbb{E}_{(X_{-K},K)\sim p_{\mathcal{X},\mathcal{K}}} \left[Cov_{X_K|(X_{-K},K)}(f, g)\right]$$

The proof of Proposition 4.1.1 is in Section 4.3.3.3.

Analogous to Theorem 4.1.2, we again show that the statistical efficiency of the adaptively-weighted MPLE (Definition 4.1.7), captured by the asymptotic variance, can be related to the Poincaré constant of the corresponding adaptively-weighted Block dynamics (Definition 4.1.8):

Theorem 4.1.3 (Asymptotic variance of adaptively-weighted MPLE under a Poincaré Inequality, generalization of Theorem 4.1.2). *Suppose the distribution p_{θ^*} satisfies a Poincaré inequality with constant C with respect to the adaptively-weighted block dynamics. Then under Assumption 4.1.1 and Assumption 4.1.2 where $p_\theta(x_K|x_{-K})$ is replaced by $p_\theta(x_K|x_{-K}, K)$, the asymptotic variance of the adaptively-weighted MPLE can be bounded as: $\Gamma_{PL} \preceq C\mathcal{I}^{-1}$ where $\mathcal{I}$ is the Fisher Information matrix (Definition 4.1.1).*

The proof of Theorem 4.1.3 is in Section 4.3.3.4.

4.1.3 Finite sample bounds and distributional distance

The framework in Section 4.1.2 was asymptotic in nature, and used parameter closeness as a notion of "quality" of the estimator. In this section, we remove both requirements, at the cost of the bounds depending on a notion of "complexity" of the parametric class we are fitting. It turns out that we can prove very similar results, with the notion of "mixing" — as captured by the Poincaré constant — being replaced by a different constant called the "approximate tensorization constant". These results mirror results in Section 5.1 in Koehler et al. (2023), who focus on 1-MPLE and use a differrent notion of "complexity" based on Rademacher complexity. We first introduce several preliminary concepts.

Definition 4.1.9 (Block approximate tensorization of entropy (Caputo & Parisi, 2021)). *Under fixed distribution $p_{\mathcal{K}}(\cdot \mid X)$ over binary masks $\mathcal{K}$ conditioned on X, we say the distribution $q_{\mathcal{X}}$ over $\mathcal{X}$ satisfies* block-generalized approximate tensorization of entropy *with constant $\bar{C}_{AT}(q_{\mathcal{X}})$ if for any distribution $r_{\mathcal{X}}$ over $\mathcal{X}$,*

$$D_{\mathrm{KL}}(r_{\mathcal{X}}, q_{\mathcal{X}}) \leq \bar{C}_{AT}(q_{\mathcal{X}}) \cdot \mathbb{E}_{X\sim r_{\mathcal{X}}}[\mathbb{E}_{K\sim p_{\mathcal{K}}(\cdot|X)}[D_{\mathrm{KL}}\left(r_{\mathcal{X}}(\cdot \mid X_{-K}, K), q_{\mathcal{X}}(\cdot \mid X_{-K}, K)\right)]]$$

This inequality is closely related to the mixing time of weighted-block dynamics (Definition 4.1.6). Namely, the inequality is weaker than the standard discrete version of the log-Sobolev inequality (Diaconis & Saloff-Coste, 1996) and stronger than the Modified Log-Sobolev Inequality (Bobkov & Tetali, 2006), which implies exponential ergodicity of the weighted block-dynamics in KL divergence[10], that is: $\mathrm{KL}(p_t, q) \leq e^{-2t/C_{AT}(q)}\mathrm{KL}(p_0, q)$.

To bound the distance between the population and empirical losses, as well as relate it to the distance between the estimated parameters and the ground truth, we first introduce a few useful pieces of notation.

[8] This is analogous to the training objective setting in Definition 4.1.7.

[9] Defined analogously as in Definition 4.1.

[10] This in turns, also implies a Poincaré inequality and exponential ergodicity in χ^2 divergence.

Notation: For each sample $X^{(i)}, i \in [n]$, we assume we observe m masks $\{K_j^{(i)} \mid j \in [m]\}$ sampled iid from $p_{\mathcal{K}}(\cdot \mid X^{(i)})$. We denote the corresponding empirical loss by

$$\hat{L}_{PL}(\theta) \coloneqq \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} -\log p_\theta(X^{(i)}_{K_j^{(i)}} | X^{(i)}_{-K_j^{(i)}}, K_j^{(i)}).$$

Furthermore, we will denote by $\tilde{p}_{\mathcal{X}}$ the uniform distribution over $\{X^{(i)}, i \in [n]\}$, and denote

$$\tilde{L}_{PL}(\theta) \coloneqq \mathbb{E}_{X \sim \tilde{p}_{\mathcal{X}}, K \sim p_{\mathcal{K}}(\cdot|X)} \left[-\log p_\theta(X_K | X_{-K}, K) \right] \tag{4.2}$$

This is an intermediate quantity: it averages in the population sense over the masks, but it assumes a finite number of samples from $p_{\mathcal{X}}$. It will be a useful intermediate quantity for several concentration bounds.

We will also need a few mild assumptions on the distribution we are fitting. First, we assume that the learned conditional probabilities are uniformly lower-bounded by a constant:

Assumption 4.1.3 (Support margin). *Exists constant $\beta \in (0, 1)$ s.t. $\forall X \in \mathcal{X}, \forall K \subset [N]$, $\forall \theta \in \Theta$, if $p_{\mathcal{X}}(X_K | X_{-K}, K) > 0$, then $p_\theta(X_K | X_{-K}, K) \geq \beta$.*

We also assume that the log-probabilities (and hence the losses $\hat{L}_{PL}$ and $\tilde{L}_{PL}$) are Lipschitz with respect to θ, and Θ has a finite covering bound. Namely:

Assumption 4.1.4 (Covering bound and Lipschitzness). *$\forall \epsilon > 0$, there exists a finite partition $Par_\epsilon(\Theta) = \{\Theta_1, \cdots, \Theta_{|Par(\Theta)|}\}$ of Θ, such that $\forall i, \forall \theta_1, \theta_2 \in \Theta_i$, and $\forall (X, K) \in \mathcal{X} \times \mathcal{K}$:*

$$|\log p_{\theta_1}(X_K | X_{-K}, K) - \log p_{\theta_2}(X_K | X_{-K}, K)| \leq \frac{\epsilon}{2}.$$

Moreover, $C_\epsilon(\Theta)$ denote the smallest possible cardinality among such partitions $Par_\epsilon(\Theta)$.

With this setup, we can prove the following finite-sample bound on the closeness of the learned distribution, provided the weighted pseudolikelihood loss (Definition 4.1.2) is small:

Theorem 4.1.4 (Generalization bound for learning the joint distribution). *Let $\hat{\theta} \coloneqq \arg\min_\theta \hat{L}_{PL}(\theta)$. Under Assumption 4.1.3 and Assumption 4.1.4, $\forall \epsilon > 0$, $\forall \delta \in (0, 1)$, with probability at least $1 - \delta$ we have $D_{\mathrm{TV}}\left(p_{\hat{\theta}}, p_{\mathcal{X}}\right) < \sqrt{\frac{1}{2} \bar{C}_{AT}(p_{\hat{\theta}}) \left(\hat{L}_{PL}(\hat{\theta}) + B \cdot \ln \frac{1}{\beta} + \epsilon \right) + C}$ where $B = \sqrt{\frac{2^{3N} |\Omega|^N C_\epsilon(\Theta)}{m \cdot \delta}} + \sqrt{\frac{\ln \frac{8 C_\epsilon(\Theta)}{\delta}}{2n}}$, and $C = \sqrt{\frac{|\Omega|^{3N}}{8 \delta n}}$.*

Proof of Theorem 4.1.4 is in Section 4.3.4. We can compare the statement to Theorem 4.1.3: (1) On the LHS, rather than parameter distance, we have total variation distance between the learned distribution and p. (2) On the RHS, rather than a Poincaré inequality, we have the $\bar{C}_{AT}(p_{\hat{\theta}})$ constant. (3) On the RHS, instead of the Fisher information matrix, we have quantities capturing the generalization error, through a notion of complexity of the class ($C_\epsilon(\Theta)$).

4.1.4 Inference-time limitations due to parallelism

In this section, we focus on limitations in the representational and computational efficiency that arise when using a parallel decoding approach to implement a step of the inference-time sampling algorithm. Precisely, at inference-time, using weighted block-dynamics with bigger blocks enables larger sets of coordinates to be re-randomized, facilitating a faster mixing time. A canonical example of this are k-Gibbs samplers:

Definition 4.1.10 (k-Gibbs sampler). *The k-Gibbs sampler is a special case of the block dynamics (Definition 4.1.6) when $\mathcal{K} \coloneqq \{K \subseteq [N] \mid |K| = k\}$, and $p_{\mathcal{K}} = Unif(\mathcal{K})$.*

$$X_K^{(t+1)} \sim p\left(\cdot \mid X_{-K}^{(t)}\right), \; X_{-K}^{(t+1)} = X_{-K}^{(t)} \tag{4.3}$$

Samplers with larger k are well-known to mix faster (e.g., Lee (2023) shows the Poincaré inequality improves by a factor of at least k). However, taking a step of this Markov Chain requires being able to re-randomize the k coordinates according to their conditional distribution — which is intuitively harder for larger k if we are trying to re-randomize the coordinates in parallel.

In Section 4.1.4.1, we show that the canonical GMLM-type parallel decoding language models can only implement Markov chains whose transitions are *product distributions* over the sequence positions. [11] We also consider a natural Markov Chain whose transitions are product distributions, and show it can be substantially slower to reach the modes of the distribution compared to k-Gibbs for a large k (i.e. a Markov Chain with transitions that are far from a product distribution). Precisely, we consider:

Definition 4.1.11 (Independent parallel sampler)**.** *The* independent parallel sampler *performs* coordinate-wise *updates for all i in parallel* [12]*, namely:*

$$\forall i \in [N],\ X_i^{(t+1)} \sim p\left(\cdot \mid X_{-\{i\}}^{(t)}\right) \tag{4.4}$$

In Section 4.1.4.2, we show that even if we do not care about mixing—just reaching the modes of the distribution—the independent parallel sampler can be much slower compared to k-Gibbs, for a large k.

4.1.4.1 Which Markov Chains are implementable via parallel decoding?

In this section, we characterize the power and restrictions of Transformers at inference time when they are restricted to decoding the tokens of the sequence in parallel. The inference algorithms for a model that has access to approximate conditional probabilities typically look like (potentially multiple) steps of block dynamics (Definition 4.1.6). We focus on understanding what kinds of transitions are implementable with a standard Transformer architecture.

Note that while there are well-known prior results about the expressive power of Transformers as sequence-to-sequence modelers (Yun et al., 2020), representing steps of a Markov Chain with parallel decoding is more subtle, due to the fact that a step of a Markov Chain requires randomness. First, we state a result characterizing the power of Transformers to approximate "deterministic" Markov Chains: that is, Markov Chains whose transition distributions are delta functions. Unsurprisingly, standard universal approximation results can be adapted easily to this case. Namely:

Proposition 4.1.2 (informal)**.** *Transformers (with sufficient depth and width) can implement any number of transitions of any deterministic Markov Chain over sequences in Ω^N.*

On the other hand, Transformers using parallel decoding cannot implement general Markov chains over Ω^N. In fact, they can only implement Markov Chains for which the transition probabilities are product distributions:

Proposition 4.1.3 (informal)**.** *The class of Markov chains over sequences in Ω^N implementable by (sufficiently wide and deep) Transformers is those whose next-state transition probability distributions are product distributions over the positions, conditioned on the current state.*

Background information on the Transformer architecture, as well as proofs of formalized versions of Proposition 4.1.2 and Proposition 4.1.3 are relegated to Section 4.3.9. Note that this does *not* mean one can only simulate Markov Chains whose *stationary* distribution is a product distribution. In fact, the standard 1-Gibbs sampler, by virtue of the fact that it only updates one coordinate at a time, encodes a product distribution for each transition. On the other hand, under fairly mild conditions on a joint p, the 1-Gibbs sampler corresponding to p is ergodic and has p as a stationary distribution. On the other hand, a step of a k-Gibbs sampler for $k > 1$ is in general *not* a product distribution, and will not be implementable by a Transformer with parallel decoding.

[11] Remark 4.3.4 in Section 4.3.9 connects our results to technical details of model architectures in prior works.

[12] The stationary distribution of this chain is unclear: in fact, it is not even clear the chain is ergodic.

4.1.4.2 Markov Chains with dependent transitions can be (much) faster

In this section, we show that the k-Gibbs (Definition 4.1.10)—the prototypical example of a Markov Chain with dependent transitions—can reach modes of the distribution much faster than the independent parallel sampler (Definition 4.1.11)—the prototypical example of a Markov Chain with independent transitions. Intuitively, in cases where there is a strong dependence between subsets of variables, jointly updating them will bring us much faster to their modes.

The toy probabilistic family in which we will illustrate this phenomenon is *Ising models*, a canonical example of an undirected model in which "dependence" between the variables can be easily modulated. Precisely:

Definition 4.1.12 (Ising models)**.** *An* Ising model *is a distribution* $p : \{\pm 1\}^N \to \mathbb{R}^+$*, defined by a graph* $G = (V, E)$ *with* $|V| = N$ *and parameters* $\left\{\boldsymbol{J}_{\{i,j\}} : \{i,j\} \in E\right\}$ *and* $\boldsymbol{h} \in \mathbb{R}^N$ *such that:*

$$p_G(X = x) \propto \exp\left(\sum_{i\in[N]} \boldsymbol{h}_i x_i + \sum_{\{i,j\}\in E} \boldsymbol{J}_{\{i,j\}} x_i x_j\right), \tag{4.5}$$

For the results in this section, we will consider a graph G that consists of a union of a clique C_G (in which $|C_G| \geq 2$, and the pairwise interactions among the variables are strong) and a set of $N - |C_G|$ independent vertices. More formally, we consider:

$$p_G(X = x) \propto \exp\left(\sum_{i\in[N]} \boldsymbol{h}_i x_i + \sum_{i\neq j, i,j\in C_G} J x_i x_j\right) \tag{4.6}$$

such that $\sum_{i\in C_G} \boldsymbol{h}_i > 0$ and $J > 0$. This is a *ferromagnetic* Ising model (i.e. the pairwise interactions prefer the variables to have the same value). Moreover, when $J \gg \|\boldsymbol{h}\|_1$, the distribution p_G has two "modes", in which all variables in C_G have the same value:

$$\mathcal{R}_1 \coloneqq \{x \in \{-1,1\}^N | x_i = 1\, \forall i \in C_G\} \tag{4.7}$$

$$\mathcal{R}_{-1} \coloneqq \{x \in \{-1,1\}^N | x_i = -1\, \forall i \in C_G\} \tag{4.8}$$

The above distribution can be seen as a toy model of language tasks in which grammatical rules or semantic constraints create "clusters" of positions in which changing isolated words leads to very unlikely sentences. Next, we formalize the concentration around the "modes":

Assumption 4.1.5 (Strongly ferromagnetic Ising model)**.** *There exist constants* $h_G > 0, J_0 > 0$ *such that* $h_G \coloneqq \sum_{i\in C_G} \boldsymbol{h}_i > \sum_{i\notin C_G} |\boldsymbol{h}_i|$, $J - \|\boldsymbol{h}\|_1 \geq J_0$.

Informally, under Assumption 4.1.5, sequences in $\mathcal{R}_1$ are much more likely under the groundtruth distribution than those in $\mathcal{R}_{-1}$, which are further much more likely than all other sequences. The formal statement and proof are in Section 4.3.5. As a result, we can think of sampling from $\mathcal{R}_1$ as analogous to sampling a high-quality sentence, and moreover, not reaching $\mathcal{R}_1$ implies the Markov chain sampling process has not mixed to the groundtruth distribution yet. Continuing the analogy to language tasks, in tasks like machine translation, for each source sentence, sampling one high-quality target sentence is potentially good enough. In some other tasks like creative writing, producing well-calibrated samples might be desirable—so mixing would be needed.

First, we show that running the **k-Gibbs sampler** requires a small number of steps to reach $\mathcal{R}_1$. This implies that if a model can efficiently approximate one step of **k-Gibbs sampler**, then it is fast to sample a high-probability sequence by iteratively applying the model. Proof is in Section 4.3.6.

Proposition 4.1.4 (k-Gibbs sampler can reach the mode fast)**.** *Consider the Ising model in Equation* (4.6) *satisfying Assumption 4.1.5. Let us denote* $(0,1) \ni c_{\mathcal{R}_1} := 1 - \frac{\binom{N-|C_G|}{k-|C_G|}}{\binom{N}{k}} \frac{e^{2(J_0+h_G)}}{e^{2(J_0+h_G)}+e^{2J_0}+2^{|C_G|}-2}$.

Then, for any initial $\boldsymbol{X}^{(0)}$ *and* $\delta \in (0,1)$*, with probability at least* $1-\delta$*, after* $T := \left\lceil \log_{c_{\mathcal{R}_1}} \delta \right\rceil$ *steps of* ***k*-Gibbs sampler** *(Definition 4.1.10) with* $k \geq |C_G|$*, we have* $\{\boldsymbol{X}^{(t)} | t \in [T]\} \cap \mathcal{R}_1 \neq \emptyset$*.*

By contrast, we show that for nontrivial probability over the randomness in the initial sequence, running **independent parallel** requires a large number of steps to reach the largest mode $\mathcal{R}_1$ of the distribution, which implies that the sampling process may not quickly reach a high-probability sequence.

Proposition 4.1.5 (Independent parallel sampling stuck in bad samples)**.** *Consider the Ising model in Equation* (4.6) *satisfying Assumption 4.1.5. Let us denote* $c_{stuck} := \frac{2\left(-1+\frac{1-\exp(-2J_0)}{\exp(-2J_0)+1}\frac{|C_G|}{2}\right)^2}{|C_G|}$.

For an initial $\boldsymbol{X}^{(0)}$ *such that* $\sum_{i \in C_G} \boldsymbol{X}_i^{(0)} \leq -2$*, for any* $\delta \in (0,1)$*, with probability at least* $1-\delta$*, after* $T := \left\lfloor \frac{\delta}{2} \exp(c_{stuck}) \right\rfloor$ *steps of* ***independent parallel*** *(Definition 4.1.11), we have* $\forall t \in [T], \sum_{i \in C_G} \boldsymbol{X}_i^{(t)} \leq -2$*.*

The proof is in Section 4.3.7. Combining Proposition 4.1.4 and Proposition 4.1.5 leads to a separation result between ***k*-Gibbs sampler** and **independent parallel**, in particular when the clique size in G is large and dependency is strong within the clique: with high probability, while the former reaches $\mathcal{R}_1$ in 1 step, the latter cannot do so in arbitrarily large number of steps:

Assumption 4.1.6 (Strong interactions in Ising model)**.** *On Ising model* G *in Equation* (4.6)*, for parameters* $\delta \in (0,1)$ *and* $M \in \mathbb{N}_+$*,*

$$
\begin{aligned}
|C_G| &\geq 8\left(1+\ln\frac{4M}{\delta}\right) \\
h_G &\geq \frac{1}{2}\ln\frac{2(4-\delta)}{\delta} \\
J_0 &\geq \frac{1}{2}|C_G|\ln 2
\end{aligned}
$$

Assumption 4.1.7 (Large coordinate set per update)**.** *When running the* k*-Gibbs sampler (Definition 4.1.10) on Ising model* G *in Equation* (4.6)*, we assume* k *is large wrt parameter* $\delta \in (0,1)$*:*

$$
k \geq \max\{|C_G|, N - \delta\frac{N+1}{(4-\delta)|C_G|+\delta}\}
$$

Remark 4.1.4. *Assumption 4.1.7 requires* k *to be not much smaller than* N*. When* N *is small and* $\delta \approx 0$*, Assumption 4.1.7 essentially requires* $k = N$*. When* $N \gg |C_G|$*, Assumption 4.1.7 allows a larger gap* $N-k$*.*

Corollary 4.1.1 (Separation between N-Gibbs sampler and independent parallel sampling)**.** *On Ising model* G *in Equation* (4.6) *under Assumption 4.1.5,* $\forall \delta \in (0,1)$*,* $\forall M \in \mathbb{N}_+$*, If* G *additionally satisfies Assumption 4.1.6 and Assumption 4.1.7 and the initial* $\boldsymbol{X}^{(0)}$ *is such that* $\sum_{i \in C_G} \boldsymbol{X}_i^{(0)} \leq -2$*, then with probability at least* $1-\delta$*,*

1. *Running the* ***k*-Gibbs sampler***:* $\boldsymbol{X}_{k\text{-}Gibbs}^{(1)} \in \mathcal{R}_1$*, and*
2. *Running* ***independent parallel****:* $\{\boldsymbol{X}_{indep}^{(t)} | t \in [M]\} \cap \mathcal{R}_1 = \emptyset$

Proof is in Section 4.3.8. We empirically verify our theory in Section 4.2.1.2.

4.2 Experiments

4.2.1 Synthetic experiments on Ising model

To empirically validate Theorem 4.1.1 (Masking more is (statistically) better), we run controlled experiments with synthetic data generated by a ground-truth Ising model. We train an Ising model using k-pseudolikelihood, and measure the squared error of parameter estimation. The results verify that with the same training data size, larger k leads to lower error. We plot the results in Figure 4.1, with several related experiments, in Section 4.2.1.1 and Section 4.2.1.2. [13]

4.2.1.1 Masking more is statistically better for learning synthetic Ising models

We show our observations in Section 4.2.1 and Figure 4.1 are robust to the shape of the groundtruth Ising model distribution: under a much more peaky groundtruth distribution (with 2 modes), it still holds that with the same training data size, larger k leads to lower error. We plot the results in Figure 4.2.

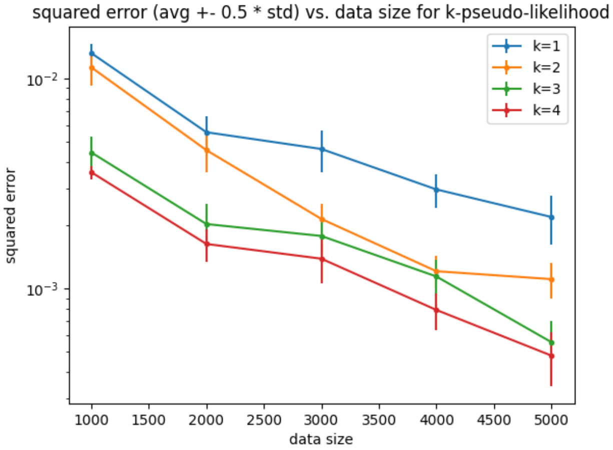


Figure 4.1: Average squared error in parameter estimation for fitting an Ising model on data generated by a groundtruth Ising model ($N = |C_G| = 4, J = 0.05, h_i = 0$ in Equation (4.6)) using the k-pseudolikelihood objective optimized by gradient descent. Error bars denote ± 0.5 * stdev for 10 repetitions of the experiment.

[13] Related simulations were also reported in Huang & Ogata (2002).

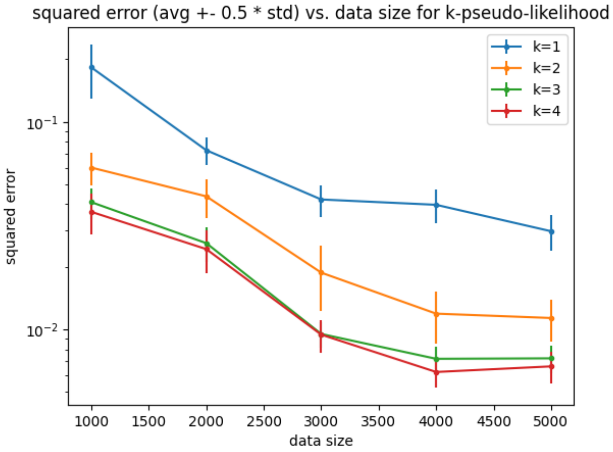


Figure 4.2: Average squared error in parameter estimation for fitting an Ising model on data generated by a groundtruth Ising model ($N = |C_G| = 4, J = 0.3, h_i = 0$ in Equation (4.6)) using the k-pseudolikelihood objective optimized by gradient descent. Error bars denote ± 0.5 * stdev for 10 repetitions of the experiment.

4.2.1.2 Markov Chains with dependent transitions can be (much) faster in sampling Ising models

To verify our theory in Section 4.1.4.2, we run controlled experiments benchmarking various sampling algorithms for Ising models: k-Gibbs sampler (Definition 4.1.10), and the independent parallel sampler (Definition 4.1.11).

The Ising model distribution that we sample from contains two modes, one larger and the other smaller, corresponding to $\mathcal{R}_1$ and $\mathcal{R}_{-1}$ defined in Equation (4.7) and Equation (4.8), respectively.

We show in Figure 4.3 that if we initialize the sample in the smaller mode $\mathcal{R}_{-1}$, running the k-Gibbs sampler (Definition 4.1.10) can often reach the larger mode $\mathcal{R}_1$ within a relatively small number of steps (though more peaky distributions i.e. those with larger J, are slower to sample). Moreover, larger k is faster than smaller k. By contrast, running the independent parallel sampler (Definition 4.1.11) cannot reach $\mathcal{R}_1$ within the compute budget we set. The results verify our theory in Section 4.1.4.2 that Markov Chains with dependent transitions can be (much) faster in sampling Ising models (compared with the independent parallel sampler).

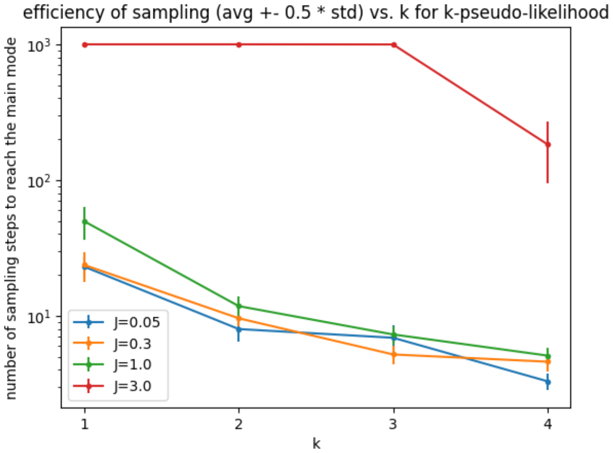


Figure 4.3: Number of steps for the k-Gibbs sampler (Definition 4.1.10) to reach the larger mode $\mathcal{R}_1$ (Equation (4.7)) of Ising models, starting from the smaller mode $\mathcal{R}_{-1}$ (Equation (4.8)). The parameters of our Ising models are: $N = 10, |C_G| = 4, h_i = 5.0$ in Equation (4.6). We vary the parameter J (a larger J corresponds to a more peaky distribution). Error bars denote ± 0.5 * stdev for 10 repetitions of the experiment. The compute budget is 1000 steps. Thus, a point with vertical coordinate 10^3 means that the sampler did not reach $\mathcal{R}_1$ within compute budget. The k-Gibbs sampler can often reach the $\mathcal{R}_1$ (larger k is faster). For context, the independent parallel sampler (not on the plot) can never reach $\mathcal{R}_1$ within the compute budget for any of the J's we tried.

4.2.2 Parallel Decoding by Iterative Refinement (PaDIR) on real language data

We consider an encoder-decoder architecture, in which the decoder is modified to be *non-autoregressive*: instead of iteratively predicting the next token, each of our decoder forward pass predicts an update to *all* target positions *in parallel*. The encoder extracts features from the source sequence, and based on these features, each decoder forward pass refines its current hypothesis of the target sequence. The initial decoder hypothesis is a purely random sequence, and more decoder forward passes correspond to more steps of refinement. [14] Note that we are *not* the first in the literature to propose this language modeling paradigm. [15] Our focus in this paper is to provide theoretical and empirical analyses to characterize its potentials, limitations and document useful training practices.

4.2.2.1 Inference approach for PaDIR

An input sequence X^{source} first goes through the encoder $f^{\text{enc}}_{\theta_e}$ (parameterized by θ_e) to produce the hidden representation h:

$$h = f^{\text{enc}}_{\theta_e}(X^{\text{source}})$$

A length predictor $f^{\text{len}}_{\theta_l}$ (parameterized by θ_l) takes h and predicts B_l most likely target lengths, where $B_l \in \mathbb{N}_+$ (beam size for length prediction) is an inference-time hyperparameter.

For each predicted length N, an initial hypothesis target sequence $X^{(0)} = X^{(0)}_1 \cdots X^{(0)}_N$ in which each $X^{(0)}_i$ can be a `[MASK]` token, or chosen uniformly randomly from the vocabulary of tokens.

For each decoder step $t \in 1 \cdots T$, the decoder $f^{\text{dec}}_{\theta_d}$ (parameterized by θ_d) takes two inputs: h and $X^{(t)}_{1 \cdots N}$, and refines the hypothesis target sequence to $X^{(t+1)}_{1 \cdots N}$, using one forward pass:

$$X^{(t+1)}_{1 \cdots N} = f^{\text{dec}}_{\theta_d}(X^{(t)}_{1 \cdots N}, h) \tag{4.9}$$

where $T \in \mathbb{N}_+$ (number of refinement steps) is an inference-time hyperparameter, and we can stop early if $X^{(t+1)} = X^{(t)}$.

4.2.2.2 Training approach for PaDIR

One-stage training Given source sequence X^{source} and target sequence X^{target} in the supervised training data $\mathcal{D}_{\text{train}}$, we use a preprocessing rule to create the initial hypothesis target sequence $X^{(0)}$. [16] The training objective is

$$L^{(1)} = \sum_{X^{\text{source}}, X^{\text{target}} \in \mathcal{D}_{\text{train}}} l(f^{\text{dec}}_{\theta_d}(X^{(0)}, f^{\text{enc}}_{\theta_e}(X^{\text{source}})) \tag{4.10}$$

where l is the cross-entropy loss applied to each position.

Multi-stage training One limitation of the one-stage training is that the inference situation is *out-of-distribution*: when decoder step $t > 1$, the model needs to refine its own predictions in step $t - 1$, which is not reflected in the training objective. Therefore, we use the multi-stage training objective (Ghazvininejad et al., 2020; Savinov et al., 2022): $L^{(S)} = \frac{1}{S} \sum_{s \in [S]} L^{(s)}$ where S is the number of training stages, and $L^{(s)} = \sum_{X^{\text{source}}, X^{\text{target}} \in \mathcal{D}_{\text{train}}} l(f^{\text{dec}}_{\theta_d}(X^{(s-1)}, f^{\text{enc}}_{\theta_e}(X^{\text{source}}))$

[14] In principle, following a similar paradigm, a non-autoregressive *decoder-only* architecture is also possible. In this work we use encoder-decoder for two reasons: (1) Efficiency: in the iterative refinement process of the hypothesis target sequence, each forward pass only involves the decoder, but not the encoder. (2) Benchmarking: the encoder-decoder design is closer to a series of prior works, allowing for more informative comparison on benchmarks.

[15] Representative prior works: Ghazvininejad et al. (2020); Savinov et al. (2022), inter alia. See Section 4.4.

[16] Each position in $X^{(0)}$ may contain a `[MASK]` token, a random token, or the correct token in X^{source}, dependning on the preprocessing rule.

4.2.3 Dataset and evaluation

We train models on machine translation datasets, provide practical recommendations based on our empirical observations, and discuss their connections to our theory.

Model training We use Transformer encoder-decoder with size similar to Transformer-Base (Vaswani et al., 2017) and T5-Small-1.0 (Raffel et al., 2020): 6 encoder and decoder layers, 8 attention heads, 512 embedding dimensions and 2048 FFN hidden dim. We add a positional attention mechanism (Gu et al., 2018; Kreutzer et al., 2020) in each Transformer layer and use learnt positional embeddings. The total number of parameters is 67M. We initialize model parameters randomly and train using a batch size of 2048 for 500k iterations, with a 10% dropout rate, 15% unmasking rate [17] and 2 training stages. The optimizer is AdaFactor (Shazeer & Stern, 2018), with default T5X hyperparameters (Roberts et al., 2022). The learning rate peaks at 0.003 with a linear rampup for 10k steps followed by cosine decay, from and to a minimum value of $1e-5$. Unlike most prior work, we do not use a remasking schedule; [18] we simply remask token-level stutter (i.e., consecutive repeated tokens) across iterations and drop repeated tokens after the final iteration. As commonly done, we distill our models by training on the output of an autoregressive model. For simplicity, we use the Google Cloud Translation API to generate this distillation data.

Datasets We evaluate our models on machine translation benchmarks commonly used in the non-autoregressive modeling literature. We conduct experiments on both directions of three WMT datasets: WMT14 DE↔EN ($4.5M$ examples) (Bojar et al., 2014), WMT16 RO↔EN ($610k$ examples) (Bojar et al., 2016) and WMT17 ZH↔EN ($20M$ examples) (Bojar et al., 2017). We load the data from the `tensorflow_datasets` library and do not apply any preprocessing other than sentence piece tokenization (Kudo & Richardson (2018)). Bilingual vocabularies of 32k tokens are created using the training sets of each language pair.

Benchmarking PaDIR models and AR models reach similar BLEU (Papineni et al., 2002) and BLEURT (Sellam et al., 2020; Pu et al., 2021) scores. Quantitative experimental results and common baselines are shown in Table 4.1, Table 4.2, and Table 4.3. We first present these quantitative metrics, and then discuss several considerations that we put into selecting those evaluation metrics.

While bridging the gap between autoregressive and non-autoregressive model has so far focused on achieving parity in terms of BLEU scores, we believe this is insufficient. Since BLEU relies on n-gram overlaps between groundtruths and model predictions, it does not capture readability very well. Yet readability is paramount for most practical applications, and it is indisputably something that current autoregressive LMs excel at. To provide additional perspectives, we introduce a word-level stutter metric, computing how often consecutive words are repeated in the model output but not in the reference. For all datasets, we found that word-level stutter is 2 or more times more frequent for non-autoregressive models.

Discussion on metrics We measure BLEU (Papineni et al., 2002) using the SacreBLEU implementation (Post, 2018) with language appropriate tokenizers [19]. For the same model, SacreBLEU on average reports a lower score than BLEU (e.g. see Savinov et al. (2022)). Unfortunately, this does not allow a direct comparison with most of the existing literature. This is a deliberate choice since it has been shown that subtle differences in preprocessing can significantly impact metrics (Schmidt et al., 2022), making comparisons error prone, and SacreBLEU is the recommended metric in Post (2018). Furthermore, common preprocessing steps (lowercasing, separating punctuation, stripping diacritics, etc.) may artificially inflate scores while not being fully reversible, as such preventing real-world uses for such models.

[17] This means, in Equation (4.10), 15% of the tokens in $X^{(0)}$ are the correct tokens in X^{target}, and the remaining 85% are random tokens in the vocabulary.

[18] We experimented with various remasking schedules but the results were not visibly affected.

[19] For public reproducibility: SacreBLEU signatures: BLEU+c.mixed+#.1+s.exp+tok.zh+v.1.3.0 for Chinese and BLEU+c.mixed+#.1+s.exp+tok.13a+v.1.3.0 for other languages.

Table 4.1: Test SacreBLEU scores on three WMT datasets. We report scores without any preprocessing. Our AR baselines are trained on the distilled dataset for a fair comparison. The 'Steps' column indicates the number of decoding iterations. The '# Hyp.' column denotes the number of hypotheses decoded in parallel (beam size for AR models and top_k predicted lengths for NAR models).

			WMT14		WMT16		WMT17	
Model	# Hyp.	Steps	DE→EN	EN→DE	RO→EN	EN→RO	ZH→EN	EN→ZH
AR Baselines	5	N	33.50	29.54	34.89	29.75	27.59	33.94
PaDIR	5	4	33.49	28.61	33.98	28.98	26.47	32.59
	5	10	33.63	28.58	33.99	28.97	26.54	32.68

Table 4.2: Test BLEU scores on three WMT datasets for baselines. Note that they use different BLEU implementations and sometimes additional preprocessing than the results reported for our approach. We include results for our PaDIR under the T5X default BLEU score (SacreBLEU tok_intl). As we remarked in the "Discussion on metrics" part of Section 4.2.3, these different BLEU implementations may not be directly comparable.

			WMT14		WMT16		WMT17	
Model	# Hyp.	Steps	DE→EN	EN→DE	RO→EN	EN→RO	ZH→EN	EN→ZH
DisCo AR Baselines	5	N	31.71	28.60	34.46	34.16	24.65	35.01
CMLM	5	4	30.75	26.73	33.02	33.67	22.57	33.58
	5	10	31.24	27.39	33.67	33.33	23.76	34.24
DisCo Easy-First	5	3-6	31.31	27.34	33.25	33.22	23.83	34.63
SUNDAE Stochastic	16	4	32.10	27.94	-	-	-	-
	16	10	32.29	28.33	-	-	-	-
PaDIR	5	4	34.17	29.49	34.55	29.57	27.18	32.59
	5	10	34.33	29.48	34.57	29.56	27.25	32.60

Table 4.3: Test BLEURT scores on three WMT datasets for our models.

			WMT14		WMT16		WMT17	
Model	# Hyp.	Steps	DE→EN	EN→DE	RO→EN	EN→RO	ZH→EN	EN→ZH
AR Baselines	5	N	73.55	74.97	67.23	71.76	68.14	65.71
PaDIR	5	4	71.26	72.08	65.90	70.23	65.16	63.95
	5	10	71.82	73.28	66.09	70.49	66.19	64.30

Speed The average target length in all datasets ranges between 28 and 33 tokens, including the EOS token. As such a non-autogressive model using 4 decoding steps does 7 to 8 times fewer decoder passes. In practice we see an end-to-end speedup greater than >2x for the median and >5x for the 99th percentile latency on TPU v3 (with 4 decoding steps and batch size 1). The gap between expected and observed speedup is due to fixed costs (input tokenization, encoding, etc.) as well as a better optimization of AR decoding (e.g. through caching of intermediate results). For longer sequences, the constant number of decoding passes in GMLM is advantageous. For completeness, it is worth noting that the number of decoder passes necessary to achieve good quality (and thus model speed) is application dependent, with some tasks like non-autoregressive text in-painting remaining slower than their autoregressive counterparts, as shown in Savinov et al. (2022).

4.2.4 Connecting to theory: quantifying dependency via attention scores

Our theory suggests that stronger dependency between target positions leads to worse generalization guarantee and sampling efficiency. However, it is unclear how to measure such dependency for Transformer-based language models trained on natural language data. In this section, we empirically investigate: *how to predict what target positions have strong dependency which may be challenging for Transformers?* We test the following two hypotheses: (1) Strongly dependent target positions have larger **decoder self-attention** between each other. (2) Strongly dependent target positions have similar **cross-attention** distribution to source tokens.

For a pair of target positions, to measure how well their dependency is modeled in the generated output, we focus on adjacent repetitive tokens, a.k.a. *stutter*. Stuttering is a common error mode among parallel decoding models, and we use it as one reasonable proxy for measuring failures in modeling target-side dependency. [20] We show:

- Hypothesis 1 is unlikely to hold: even on average, stuttering positions do not have larger **decoder self-attention** between each other, compared with non-stuttering adjacent positions. [21]
- By contrary, Hypothesis 2 is potentially promising: with various of distribution distance measures, stuttering positions in the generated output have more similar **cross-attention** distributions to source tokens, compared with non-stuttering adjacent positions.

Details are in Table 4.4 and Table 4.5. There are other error modes connected to the challenge of modeling target-side dependency, but they are more ambiguous for measuring and exactly locating. We do not aim to develop decoding algorithms tailored to just reducing stuttering rate. (After all, stuttering can be easily removed by rule-based postprocessing.) Instead, the above are general-purpose hypotheses which are potentially also predictive of other (more complex) failure modes related to target-side dependency.

[20] There are other error modes connected to the challenge of modeling target-side dependency, but they are more ambiguous for measuring and exactly locating. We do not aim to develop decoding algorithms tailored to just reducing stuttering rate. (After all, stuttering can be easily removed by rule-based postprocessing.) Instead, the above are general-purpose hypotheses which are potentially also predictive of other (more complex) failure modes related to target-side dependency.

[21] Since all stuttering positions are by definition adjacent, we think a fair comparison should only consider adjacent positions for non-stuttering position pairs.

Table 4.4: Stuttering positions have comparable average last-layer self-attentions compared with non-stuttering adjacent positions. For each pair of adjacent positions in the generated sequence: (1) the 'self-attention scores' include both directions ; (2) The column 'min' denotes only including the minimum among such score over all attention heads, and likewise for 'avg' and 'max'; (3) the entries are mean ± standard deviation; (4) $\mathbb{P}\{\text{top-}k \text{ overlap}\}$ denotes the chances that the self-attention distribution at one position includes the other position among its top-k "most attended to" positions.

	self-attention scores			$\mathbb{P}\{\text{top-}k \text{ overlap}\}$	
stutter	min	avg	max	$k = 1$	$k = 2$
yes	0.0004 ± 0.0007	0.032 ± 0.023	0.16 ± 0.11	0.20	0.39
no	0.0005 ± 0.0007	0.033 ± 0.025	0.17 ± 0.12	0.17	0.37

Table 4.5: Stuttering positions on average have more similar last-layer cross-attentions than non-stuttering adjacent positions. For each pair of adjacent positions in the generated sequence: (1) the 'total variation distance' and 'cosine distance' (both have range $[0, 1]$) are taken for the two corresponding cross-attention distributions; (2) The column 'min' denotes only including the minimum among such distance over all attention heads, and likewise for 'avg' and 'max'; (3) the entries are mean ± standard deviation; (4) $\mathbb{P}\{\text{top-}k \text{ overlap}\}$ denotes the chances that the two cross-attention distributions overlap in terms of their top-k "most attended to" source positions.

	total variation distance			cosine distance			$\mathbb{P}\{\text{top-}k \text{ overlap}\}$	
stutter	min	avg	max	min	avg	max	$k = 1$	$k = 2$
yes	0.06 ± 0.05	0.13 ± 0.09	0.23 ± 0.15	0.01 ± 0.01	0.10 ± 0.06	0.25 ± 0.11	0.57	0.89
no	0.11 ± 0.10	0.23 ± 0.14	0.35 ± 0.18	0.04 ± 0.08	0.20 ± 0.11	0.38 ± 0.12	0.40	0.81

4.3 Proofs

4.3.1 Proof of Lemma 4.1.2: Generalized information matrix equality

For convenience, we restate the generalized information matrix equality we are going to show:

Lemma 4.1.2 (Generalized information matrix equality)**.** *Under Assumption 4.1.1 and Assumption 4.1.2, the weighted pseudolikelihood loss (Definition 4.1.2) verifies:* $\nabla_\theta^2 L_{PL}(\theta^*) = \mathrm{Cov}_{X\sim p_{\mathcal{X}}, K\sim p_{\mathcal{K}}}(-\nabla_\theta \log p_\theta(X_K|X_{-K}))_{|\theta=\theta^*}$.

Proof. All the expectations in the proof will be taken with respect to $(X, K) \sim p_{\mathcal{X}} \times p_{\mathcal{K}}$. To decrease the notational load, we will not explicitly write $p_{\mathcal{X}} \times p_{\mathcal{K}}$. The proof proceeds by first exchanging the order of expectations and derivatives, and using that to show the appropriate terms in the expression for $\nabla_\theta^2 L_{PL}(\theta^*)$ vanish.

Step 1: Changing the order of expectations and derivatives

We will show that the following two equalities hold:

$$\nabla_\theta \mathbb{E}_{(X,K)} \log p_\theta(x_K|x_{-K}) = \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K}) \tag{4.11}$$

$$\nabla_\theta^2 \mathbb{E}_{(X,K)} \log p_\theta(x_K|x_{-K}) = \mathbb{E}_{(X,K)} \nabla_\theta^2 \log p_\theta(x_K|x_{-K}) \tag{4.12}$$

Since Ω, $[N]$, and $K \subset [N]$ are both discrete finite, the conditions for the Dominated Convergence Theorem holds under Assumption 4.1.1: namely, there exists a function $f : \Theta \times \Omega \times \mathcal{K} \mapsto \mathbb{R}$ such that $\forall \theta \in \Theta$, $\mathbb{E}_{(X,K)}\left[f(\theta, X, K)\right] < \infty$, $\|\nabla_\theta \log p_\theta(x_K|x_{-K})\|_2 \leq f(\theta, X, K)$, and $\|\nabla_\theta^2 \log p_\theta(x_K|x_{-K})\|_F \leq f(\theta, X, K)$.

Denoting by $\mathbf{e}_i$ the i-th standard basis vector, we have:

$$\frac{\partial}{\partial \theta_j} \mathbb{E}_{(X,K)}\left[\log p_\theta(x_K|x_{-K})\right] = \lim_{h\to 0} \frac{1}{h}\left(\mathbb{E}_{(X,K)}\left[\log p_{\theta+\boldsymbol{e}_j h}(x_K|x_{-K})\right] - \mathbb{E}_{(X,K)}\left[\log p_\theta(x_K|x_{-K})\right]\right) \tag{4.13}$$

$$= \lim_{h\to 0} \mathbb{E}_{(X,K)}\left[\frac{\log p_{\theta+\boldsymbol{e}_j h}(x_K|x_{-K}) - \log p_\theta(x_K|x_{-K})}{h}\right] \tag{4.14}$$

By the Mean Value Theorem, there exists $\xi(h) \in (0, h)$ such that

$$\frac{\log p_{\theta+\boldsymbol{e}_j h}(x_K|x_{-K}) - \log p_\theta(x_K|x_{-K})}{h} = \frac{\partial}{\partial_{\theta_j}} \log p_{\theta+\boldsymbol{e}_j \xi(h)}(x_K|x_{-K})$$

So

$$\begin{aligned}
&\frac{\partial}{\partial \theta_j} \mathbb{E}_{(X,K)}\left[\log p_\theta(x_K|x_{-K})\right] \\
&= \lim_{h\to 0}\left(\mathbb{E}_{(X,K)}\left[\frac{\partial}{\partial_{\theta_j}} \log p_{\theta+\boldsymbol{e}_j\xi(h)}(x_K|x_{-K})\right]\right) \\
&= \mathbb{E}_{(X,K)}\left[\lim_{h\to 0}\left(\frac{\partial}{\partial_{\theta_j}} \log p_{\theta+\boldsymbol{e}_j\xi(h)}(x_K|x_{-K})\right)\right] \quad \text{(Dominated Convergence Thm and Assumption 4.1.1)} \\
&= \mathbb{E}_{(X,K)}\left[\frac{\partial}{\partial_{\theta_j}} \log p_\theta(x_K|x_{-K})\right]
\end{aligned}$$

This implies that

$$\nabla_\theta \mathbb{E}_{(X,K)} \log p_\theta(x_K|x_{-K}) = \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K})$$

which proves (4.13). The proof of (4.14) follows analogously.

Step 2: Rewrite $\nabla_\theta^2 L_{PL}(\theta^*)$

$$\begin{aligned}\nabla_\theta^2 L_{PL}(\theta) &= -\nabla_\theta^2 \mathbb{E}_{(X,K)} \log p_\theta(x_K|x_{-K})_{|\theta=\theta^*} \\ &\overset{\textcircled{1}}{=} -\mathbb{E}_{(X,K)} \nabla_\theta^2 \log p_\theta(x_K|x_{-K})_{|\theta=\theta^*} \\ &\overset{\textcircled{2}}{=} \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K}) \nabla_\theta \log p_\theta(x_K|x_{-K})^\top_{|\theta=\theta^*} - \frac{\nabla_\theta^2 p_\theta(x_K|x_{-K})}{p_\theta(x_K|x_{-K})}_{|\theta=\theta^*} \\ &\overset{\textcircled{3}}{=} \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K}) \nabla_\theta \log p_\theta(x_K|x_{-K})^\top_{|\theta=\theta^*} \end{aligned} \tag{4.15}$$

where $\textcircled{1}$ follows by exchanging the order of expectation and Hessian ($S \in \mathcal{S}_k$ and $x \in \Omega$ are finite), and this is valid by **Step 1** above , $\textcircled{2}$ by an application of chain rule. The last equality $\textcircled{3}$ follows by a similar calculation as the proof of the classical information matrix equality:

$$\begin{aligned} &\mathbb{E}_{(X,K)} \frac{\nabla_\theta^2 p_\theta(x_K|x_{-K})}{p_\theta(x_K|x_{-K})}_{|\theta=\theta^*} \\ &= \mathbb{E}_K \mathbb{E}_{x_{-K}} \mathbb{E}_{x_K|x_{-K}} \frac{\nabla_\theta^2 p_\theta(x_K|x_{-K})}{p_\theta(x_K|x_{-K})}_{|\theta=\theta^*} \\ &= \mathbb{E}_K \mathbb{E}_{x_{-K}} \int \frac{\nabla_\theta^2 p_{\theta^*}(x_K|x_{-K})}{p_{\theta^*}(x_K|x_{-K})} \cdot p_{\mathcal{X}}(x_K|x_{-K}) dx_K \\ &= \mathbb{E}_K \mathbb{E}_{x_{-K}} \int \nabla_\theta^2 p_{\theta^*}(x_K|x_{-K}) dx_K \quad \text{, since } p_{\theta^*} = p_{\mathcal{X}} \text{ by Assumption 4.1.2)} \\ &= \mathbb{E}_K \mathbb{E}_{x_{-K}} \nabla_\theta^2 \int p_{\theta^*}(x_K|x_{-K}) dx_K \quad \text{, by exchanging the order of expectation and Hessian} \\ &= 0 \end{aligned}$$

where the last equality follows since $\int p_{\theta^*}(x_K|x_{-K})dx_K = 1$ (so doesn't depend on θ). Similarly, we have:

$$\begin{aligned} &\mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K})_{|\theta=\theta^*} \\ &= \mathbb{E}_K \mathbb{E}_{x_{-K}} \mathbb{E}_{x_K|x_{-K}} \frac{\nabla_\theta p_\theta(x_K|x_{-K})}{p_\theta(x_K|x_{-K})}_{|\theta=\theta^*} \\ &= \mathbb{E}_K \mathbb{E}_{x_{-K}} \int \nabla_\theta p_\theta(x_K|x_{-K}) dx_K |_{\theta=\theta^*} \\ &= \mathbb{E}_K \mathbb{E}_{x_{-K}} \nabla_\theta \int p_\theta(x_K|x_{-K}) dx_K |_{\theta=\theta^*} \\ &= 0 \end{aligned}$$

where the last equality follows since $\int p_\theta(x_K|x_{-K})dx_K = 1$ (so doesn't depend on θ). Plugging this into the definition of covariance, we have:

$$\begin{aligned} &\mathrm{Cov}(\nabla_\theta - \log p_\theta(X_K|X_{-K}))_{|\theta=\theta^*} \\ &= \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K}) \nabla_\theta \log p_\theta(x_K|x_{-K})^\top \\ &\quad - \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K}) \cdot \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K})^\top_{|\theta=\theta^*} \\ &= \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K}) \nabla_\theta \log p_\theta(x_K|x_{-K})^\top_{|\theta=\theta^*} \end{aligned} \tag{4.16}$$

The proof of the lemma thus follows because the RHS of Equation (4.16) matches that of Equation (4.15).

□

4.3.2 Proof of Theorem 4.1.1: Masking more is (statistically) better

In this Section, we provide the proof for Theorem 4.1.1.

Proof of Theorem 4.1.1. All the expectations in the proof will be taken with respect to $(X, K) \sim p_{\mathcal{X}} \times p_{\mathcal{K}}$. To decrease the notational load, we will not explicitly write $p_{\mathcal{X}} \times p_{\mathcal{K}}$. By Lemma 4.1.2, we have:

$$\nabla_\theta^2 L_{PL}^k(\theta^*) = \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K}) \nabla_\theta \log p_\theta(x_K|x_{-K})^\top_{|\theta=\theta^*} \tag{4.17}$$

Let $\mathcal{S}_k$ denote the set $\{K \subset [N] \mid |K| = k\}$. For every $T \in \mathcal{S}_{k+1}$ and $a \in T$ we have:

$$\begin{aligned}\log p(x_T|x_{-T}) &= \log p(x_S, x_a|x_{-\{S\cup a\}}) \text{ where } S \coloneqq T\backslash\{a\}\\ &= \log\left(p(x_a|x_{-\{S\cup a\}}) \cdot p(x_S|x_{-\{S\cup a\}}, x_a)\right)\\ &= \log p(x_a|x_{-\{S\cup a\}}) + \log p(x_S|x_{-S})\end{aligned}$$

Using this identity, we can write:

$$\begin{aligned}\nabla_\theta^2 L_{PL}^{k+1}(\theta^*) &= \mathbb{E}_{T\sim S_{k+1}} \mathbb{E}_{x_T, x_{-T}} \nabla_\theta \log p_\theta(x_T|x_{-T}) \nabla_\theta \log p_\theta(x_T|x_{-T})^\top_{|\theta=\theta^*}\\ &= \mathbb{E}_{S\sim S_k} \mathbb{E}_{a\notin S} \mathbb{E}_{x_S, x_a, x_{-\{S\cup a\}}} \nabla_\theta \log p_\theta(x_T|x_{-T}) \nabla_\theta \log p_\theta(x_T|x_{-T})^\top_{|\theta=\theta^*}\\ &= \mathbb{E}_{S\sim S_k} \mathbb{E}_{a\notin S} \mathbb{E}_{x_S, x_a, x_{-\{S\cup a\}}} \left(\nabla_\theta \log p_\theta(x_a|x_{-\{S\cup a\}}) + \nabla_\theta \log p_\theta(x_S|x_{-S})\right)\\ &\quad \cdot \left(\nabla_\theta \log p_\theta(x_a|x_{-\{S\cup a\}}) + \nabla \log p_\theta(x_S|x_{-S})\right)^\top_{|\theta=\theta^*}\end{aligned} \tag{4.18}$$

Let us denote:

$$\begin{aligned}A &\coloneqq \mathbb{E}_{S\sim S_k} \mathbb{E}_{a\notin S} \mathbb{E}_x \nabla_\theta \log p_\theta(x_S|x_{-S}) \cdot \nabla_\theta \log p_\theta(x_S|x_{-S})^\top_{|\theta=\theta^*}\\ B &\coloneqq \mathbb{E}_{S\sim S_k} \mathbb{E}_{a\notin S} \mathbb{E}_x \nabla_\theta \log p_\theta(x_a|x_{-\{S\cup a\}}) \cdot \nabla_\theta \log p_\theta(x_S|x_{-S})^\top_{|\theta=\theta^*}\\ C &\coloneqq \mathbb{E}_{S\sim S_k} \mathbb{E}_{a\notin S} \mathbb{E}_x \nabla_\theta \log p_\theta(x_a|x_{-\{S\cup a\}}) \cdot \nabla_\theta \log p_\theta(x_a|x_{-\{S\cup a\}})^\top_{|\theta=\theta^*}\end{aligned}$$

By expanding the previous expression, we have

$$\nabla_\theta^2 L_{PL}^{k+1}(\theta^*) = A + B + B^\top + C \tag{4.19}$$

Consider A first. Note that for a fixed $S \in S_k$, $\mathbb{E}_x \nabla_\theta \log p_\theta(x_S|x_{-S}) \cdot \nabla_\theta \log p_\theta(x_S|x_{-S})^\top$ is independent of $a \notin S$ and therefore:

$$\begin{aligned}A &= \mathbb{E}_{S\sim S_k} \mathbb{E}_x \nabla_\theta \log p_\theta(x_S|x_{-S}) \cdot \nabla_\theta \log p_\theta(x_S|x_{-S})^\top_{|\theta=\theta^*}\\ &= \nabla_\theta^2 L_{PL}^k(\theta^*) \quad \text{(by Equation (4.17))}\end{aligned}$$

Proceeding to B, for a given $S \in S_k, x_{-S}$, we have

$$\begin{aligned}&\mathbb{E}_{x_S|x_{-S}} \left[\nabla_\theta \log p_\theta(x_S|x_{-S})^\top\right]_{|\theta=\theta^*}\\ &= \int \nabla_\theta \log p_\theta(x_S|x_{-S})^\top \cdot p(x_S|x_{-S}) dx_S|_{\theta=\theta^*}\\ &= \int \frac{\nabla_\theta p_\theta(x_S|x_{-S})}{p_{\theta^*}(x_S|x_{-S})}^\top \cdot p(x_S|x_{-S}) dx_S|_{\theta=\theta^*}\\ &= \int \nabla_\theta p_\theta(x_S|x_{-S})^\top dx_S|_{\theta=\theta^*}\\ &= \nabla_\theta \int p_\theta(x_S|x_{-S})^\top dx_S|_{\theta=\theta^*} \quad \text{(valid under Assumption 4.1.1, see Step 1 in the proof of Lemma 4.1.2)}\\ &= \nabla_\theta 1 = 0\end{aligned} \tag{4.20}$$

Therefore:

$$\begin{aligned}
B &= \mathbb{E}_{S\sim S_k}\mathbb{E}_{a\notin S}\mathbb{E}_{x_S,x_a,x_{-\{S\cup a\}}}\nabla_\theta \log p_\theta(x_a|x_{-\{S\cup a\}})\cdot\nabla_\theta \log p_\theta(x_S|x_{-S})^\top|_{\theta=\theta^*}\\
&= \mathbb{E}_{S\sim S_k}\mathbb{E}_{a\notin S}\mathbb{E}_{x_a,x_{-\{S\cup a\}}}\nabla_\theta \log p_\theta(x_a|x_{-\{S\cup a\}})\cdot\mathbb{E}_{x_S}\left[\nabla_\theta \log p_\theta(x_S|x_{-S})^\top\right]|_{\theta=\theta^*}\\
&\quad \text{(valid under Assumption 4.1.1, see Step 1 in the proof of Lemma 4.1.2)}\\
&= \mathbb{E}_{S\sim S_k}\mathbb{E}_{a\notin S}\mathbb{E}_{x_a,x_{-\{S\cup a\}}}\nabla_\theta \log p_\theta(x_a|x_{-\{S\cup a\}})|_{\theta=\theta^*}\cdot 0 \quad \text{(by Equation (4.20))}\\
&= 0
\end{aligned}$$

Finally, each term $\nabla_\theta \log p_\theta(x_a|x_{-\{S\cup a\}})\cdot\nabla_\theta \log p_\theta(x_a|x_{-\{S\cup a\}})^\top \succeq 0$ therefore $C \succeq 0$. Plugging this back in (4.19), we have:

$$\nabla_\theta^2 L_{PL}^{k+1}(\theta^*) = \nabla_\theta^2 L_{PL}^{k}(\theta^*) + C \succeq \nabla_\theta^2 L_{PL}^{k}(\theta^*)$$

Consequently, by monotonicity of the matrix inverse, we have

$$\Gamma_{PL}^{k+1} = \left(\nabla_\theta^2 L_{PL}^{k+1}(\theta^*)\right)^{-1} \preceq \left(\nabla_\theta^2 L_{PL}^{k}(\theta^*)\right)^{-1} = \Gamma_{PL}^{k}$$

as we need. □

4.3.3 Generalizations for adaptive masking

In this section, we provide proofs for several of the claims in Section 4.1.2.3.

4.3.3.1 Conditioning on K

First, we clarify a slightly subtle (and counterintuitive) point stressed in Remark 4.1.3: in general, $p_{\mathcal{X}}(x_K|x_{-K}, K) \neq p_{\mathcal{X}}(x_K|x_{-K})$.

Lemma 4.3.1. *Consider $\mathcal{X} = \{(0,0),(0,1),(1,0),(1,1)\}$. There exists a distribution $p_{\mathcal{X},\mathcal{K}}$ such that $p_{\mathcal{X}}(x_K|x_{-K}, K) \neq p_{\mathcal{X}}(x_K|x_{-K})$ for some $x \in \mathcal{X}$, $K \in \mathcal{K}$.*

Proof. To define $p_{\mathcal{X},\mathcal{K}}$, it suffices to define $p_{\mathcal{X}}$ and $p_{\mathcal{K}}(\cdot|x), \forall x \in \mathcal{X}$.

$$p_{\mathcal{X}}(X) = \begin{cases} (0,0), & \text{with probability } \frac{1}{2} \\ (0,1), & \text{with probability } \frac{1}{3} \\ (1,0), & \text{with probability } \frac{1}{6} \\ (0,0), & \text{with probability } 0 \end{cases}$$

and let

$$p_{\mathcal{K}}(K \mid X = (0,0)) = \begin{cases} \{0\}, & \text{with probability } \frac{1}{2} \\ \{1\}, & \text{with probability } \frac{1}{2} \end{cases}$$
$$p_{\mathcal{K}}(K \mid X = (0,1)) = \begin{cases} \{0\}, & \text{with probability } \frac{1}{3} \\ \{1\}, & \text{with probability } \frac{2}{3} \end{cases}$$
$$p_{\mathcal{K}}(K \mid X = (1,0)) = \begin{cases} \{0\}, & \text{with probability } \frac{1}{4} \\ \{1\}, & \text{with probability } \frac{3}{4} \end{cases}$$

By multiplying $p_{\mathcal{X}}(X)$ and $p_{\mathcal{K}}(K \mid X)$, we have

$$\begin{aligned} p((0,0),\{0\}) &= \frac{1}{4}, \quad p((0,0),\{1\}) = \frac{1}{4} \\ p((0,1),\{0\}) &= \frac{1}{9}, \quad p((0,1),\{1\}) = \frac{2}{9} \\ p((1,0),\{0\}) &= \frac{1}{24}, \quad p((1,0),\{1\}) = \frac{1}{8} \end{aligned}$$

Finally, we will see that $p_{\mathcal{X}}(x_1 = 0|x_0 = 0, \{0\}) \neq p_{\mathcal{X}}(x_1 = 0|x_0 = 0)$:

$$p_{\mathcal{X}}(x_1 = 0|x_0 = 0, \{0\}) = \frac{p((0,0),\{0\})}{p((0,0),\{0\}) + p((0,1),\{0\})} = \frac{9}{13}$$
$$p_{\mathcal{X}}(x_1 = 0|x_0 = 0) = \frac{p_{\mathcal{X}}((0,0))}{p_{\mathcal{X}}((0,0)) + p_{\mathcal{X}}((0,1))} = \frac{3}{5}$$

□

Instead, by correctly marginalizing, the following equality obtains:

Lemma 4.3.2. *For any distribution $p_{\mathcal{X},\mathcal{K}}$, we have:*

$$\forall x \in \mathcal{X},\ \forall K \in \mathcal{K},\ p_{\mathcal{X}}(x_K|x_{-K}) = \mathbb{E}_{K' \sim p_{\mathcal{K}}(\cdot|x_{-K})}\left[p_{\mathcal{X},\mathcal{K}}(x_K|x_{-K}, K')\right] \tag{4.21}$$

Proof. The proof proceeds by a sequence of straightforward rewrites:

$$\begin{aligned}
p_{\mathcal{X}}(x_K|x_{-K}) &= \frac{p_{\mathcal{X}}(x_K, x_{-K})}{p_{\mathcal{X}}(x_{-K})} \\
&= \sum_{K'} \frac{p_{\mathcal{X},\mathcal{K}}(x_K, x_{-K}, K')}{p_{\mathcal{X}}(x_{-K})} \\
&= \sum_{K'} \frac{p_{\mathcal{X},\mathcal{K}}(K', x_{-K})}{p_{\mathcal{X}}(x_{-K})} \cdot \frac{p_{\mathcal{X},\mathcal{K}}(x_K, x_{-K}, K')}{p_{\mathcal{X},\mathcal{K}}(x_{-K}, K')} \\
&= \sum_{K'} p_{\mathcal{K}}(K' \mid x_{-K}) p_{\mathcal{X},\mathcal{K}}(x_K|x_{-K}, K') \\
&= \mathbb{E}_{K' \sim p_{\mathcal{K}}(\cdot|x_{-s})} \left[p_{\mathcal{X},\mathcal{K}}(x_K|x_{-K}, K') \right]
\end{aligned}$$

□

4.3.3.2 Information matrix equality for adaptive masking

We prove a more general version of Lemma 4.1.2 when $p_{\mathcal{K}}$ is allowed to depend on X, which is needed for Theorem 4.1.3. Recall from Section 4.1.2.3 that the distribution $p_{\mathcal{X},\mathcal{K}}$ is defined such that:

$$p_{\mathcal{X},\mathcal{K}}(X, K) := p_{\mathcal{X}}(X) p_{\mathcal{K}}(K|X)$$

Lemma 4.3.3 (Generalized information matrix equality, adaptive masking)**.** *Under Assumption 4.1.1 and Assumption 4.1.2, the weighted pseudolikelihood loss (Definition 4.1.7) verifies:*

$$\nabla_\theta^2 L_{PL}(\theta^*) = \mathrm{Cov}_{(X,K)\sim p_{\mathcal{X},\mathcal{K}}}(-\nabla_\theta \log p_\theta(X_K|X_{-K}, K))_{|\theta=\theta^*}.$$

Proof. All the expectations in the proof will be taken with respect to $(X, K) \sim p_{\mathcal{X},\mathcal{K}}$. To decrease the notational load, we will not explicitly write $p_{\mathcal{X},\mathcal{K}}$. Same as Lemma 4.1.2, the proof proceeds by first exchanging the order of expectations and derivatives, and using that to show the appropriate terms in the expression for $\nabla_\theta^2 L_{PL}(\theta^*)$ vanish.

In fact, it's readily seen that the proof of Step 1 in Lemma 4.1.2 (Section 4.3.1) doesn't depend on $p_{\mathcal{X},\mathcal{K}}$ being a product distribution, and the same proof applies to our setting, namely we have:

$$\nabla_\theta \mathbb{E}_{(X,K)} \log p_\theta(x_K|x_{-K}, K) = \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K}, K) \tag{4.22}$$

$$\nabla_\theta^2 \mathbb{E}_{(X,K)} \log p_\theta(x_K|x_{-K}, K) = \mathbb{E}_{(X,K)} \nabla_\theta^2 \log p_\theta(x_K|x_{-K}, K) \tag{4.23}$$

We can also rewrite the expression for $\nabla_\theta^2 L_{PL}(\theta^*)$ almost the same way we did in Step 2 in Lemma 4.1.2:

$$\begin{aligned}
\nabla_\theta^2 L_{PL}(\theta) &= -\nabla_\theta^2 \mathbb{E}_{(X,K)} \log p_\theta(x_K|x_{-K}, K)_{|\theta=\theta^*} \\
&\overset{\textcircled{1}}{=} -\mathbb{E}_{(X,K)} \nabla_\theta^2 \log p_\theta(x_K|x_{-K}, K)_{|\theta=\theta^*} \\
&\overset{\textcircled{2}}{=} \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K}, K) \nabla_\theta \log p_\theta(x_K|x_{-K}, K)^\top_{|\theta=\theta^*} - \frac{\nabla_\theta^2 p_\theta(x_K|x_{-K}, K)}{p_\theta(x_K|x_{-K}, K)}_{|\theta=\theta^*} \\
&\overset{\textcircled{3}}{=} \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K}, K) \nabla_\theta \log p_\theta(x_K|x_{-K}, K)^\top_{|\theta=\theta^*}
\end{aligned} \tag{4.24}$$

where ① follows by exchanging the order of expectation and Hessian ($S \in \mathcal{S}_k$ and $x \in \Omega$ are finite), and this is valid by **Step 1** above , ② by an application of chain rule. The last equality ③ follows by a similar calculation as the proof of the classical information matrix equality (and again, analogously to the calculation in Lemma 4.1.2):

$$\begin{aligned}
&\mathbb{E}_{(X,K)} \frac{\nabla_\theta^2 p_\theta(x_K|x_{-K},K)}{p_\theta(x_K|x_{-K},K)}_{|\theta=\theta^*} \\
&= \mathbb{E}_K \mathbb{E}_{x_{-K}|K} \mathbb{E}_{x_K|x_{-K},K} \frac{\nabla_\theta^2 p_\theta(x_K|x_{-K},K)}{p_\theta(x_K|x_{-K},K)}_{|\theta=\theta^*} \\
&= \mathbb{E}_K \mathbb{E}_{x_{-K}|K} \int \frac{\nabla_\theta^2 p_{\theta^*}(x_K|x_{-K},K)}{p_{\theta^*}(x_K|x_{-K},K)} \cdot p_{\mathcal{X},\mathcal{K}}(x_K|x_{-K},K) dx_K \\
&= \mathbb{E}_K \mathbb{E}_{x_{-K}|K} \int \nabla_\theta^2 p_{\theta^*}(x_K|x_{-K},K) dx_K \quad \text{, since } p_{\theta^*} = p_{\mathcal{X}} \text{ by Assumption 4.1.2 and Definition (4.1)} \\
&= \mathbb{E}_K \mathbb{E}_{x_{-K}|K} \nabla_\theta^2 \int p_{\theta^*}(x_K|x_{-K},K) dx_K \quad \text{, by exchanging the order of expectation and Hessian} \\
&= 0
\end{aligned}$$

where the last equality follows since $\int p_{\theta^*}(x_K|x_{-K},K)dx_K = 1$ (so doesn't depend on θ). Similarly, we have:

$$\begin{aligned}
&\mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K},K)_{|\theta=\theta^*} \\
&= \mathbb{E}_K \mathbb{E}_{x_{-K}|K} \mathbb{E}_{x_K|x_{-K},K} \frac{\nabla_\theta p_\theta(x_K|x_{-K},K)}{p_\theta(x_K|x_{-K},K)}_{|\theta=\theta^*} \\
&= \mathbb{E}_K \mathbb{E}_{x_{-K}|K} \int \nabla_\theta p_\theta(x_K|x_{-K},K) dx_K|_{\theta=\theta^*} \\
&= \mathbb{E}_K \mathbb{E}_{x_{-K}|K} \nabla_\theta \int p_\theta(x_K|x_{-K},K) dx_K|_{\theta=\theta^*} \\
&= 0
\end{aligned}$$

where the last equality follows since $\int p_\theta(x_K|x_{-K},K)dx_K = 1$ (so doesn't depend on θ). Plugging this into the definition of covariance, we have:

$$\begin{aligned}
&\operatorname{Cov}(\nabla_\theta l_{PL}(\theta^*)) \\
&= \operatorname{Cov}(\nabla_\theta - \log p_\theta(X_K|X_{-K},K))_{|\theta=\theta^*} \\
&= \mathbb{E}_{(X,K)} \nabla_\theta \log p_\theta(x_K|x_{-K},K) \nabla_\theta \log p_\theta(x_K|x_{-K},K)^\top_{|\theta=\theta^*}
\end{aligned} \tag{4.25}$$

which finishes the proof of the Lemma. □

4.3.3.3 Proof of Proposition 4.1.1: Dirichlet form for adaptive block dynamics

Proposition 4.1.1 (Dirichlet form for adaptive weighted block dynamics)**.** *The Dirichlet form corresponding to the weighted block dynamics (Definition 4.1.8) is:*

$$\mathcal{E}(f,g) = \mathbb{E}_{(X_{-K},K)\sim p_{\mathcal{X},\mathcal{K}}} \left[Cov_{X_K|(X_{-K},K)}(f,g) \right]$$

Proof. Let us denote by $\Xi := \mathcal{X} \times \mathcal{K}$, and note that Ξ is the domain for both f and g. According definition of block dynamics in Definition 4.1.6, for each pair of states $(X,K_1),(Y,K_2) \in \Xi$, the transition matrix P is:

$$P\left((X,K_1),(Y,K_2)\right) = \mathbb{1}_{K_1=K_2} \mathbb{1}_{X_{-K_1}=Y_{-K_1}} p(Y_{K_1} \mid X_{-K_1}, K_1) \tag{4.26}$$

The rest of the proof is straightforward calculation, expanding the expression in the definition of the Dirichlet form (Definition 4.1.4). Namely, we have:

$$
\begin{aligned}
&\mathcal{E}_P(f,g) \\
&= \frac{1}{2} \sum_{(X,K_1),(Y,K_2)\in\Xi} p(X,K_1)P\left((X,K_1),(Y,K_2)\right)\left(f(X,K_1)-f(Y,K_2)\right)(g(X,K_1)-g(Y,K_2)) \\
&= \frac{1}{2} \sum_{(X,K_1),(Y,K_2)\in\Xi} p(X,K_1)\cdot(f(X,K_1)-f(Y,K_2))(g(X,K_1)-g(Y,K_2)) \\
&\quad \cdot \mathbb{1}_{K_1=K_2}\mathbb{1}_{X_{-K_1}=Y_{-K_1}}p(Y_{K_1}\mid X_{-K_1},X_1) \\
&= \frac{1}{2}\sum_{X\in\mathcal{X}} p_{\mathcal{X}}(X)\sum_{K\in\mathcal{K}} p_{\mathcal{K}}(K\mid X) \sum_{Y\in\mathcal{X},Y_{-K}=X_{-K}} p(Y_K\mid X_{-K},K)\cdot(f(X,K)-f(Y,K))(g(X,K)-g(Y,K)) \\
&= \frac{1}{2}\mathbb{E}_X\mathbb{E}_{K|X}\mathbb{E}_{Y_K|X_{-K},K}(f(X,K)-f(Y,K))(g(X,K)-g(Y,K)) \\
&= \frac{1}{2}\cdot\big(2\cdot\mathbb{E}_K\mathbb{E}_{X_{-K}|K}\mathbb{E}_{X_K|X_{-K},K}f(X,K)g(X,K)-2\cdot\mathbb{E}_K\mathbb{E}_{X_{-K}|K}\mathbb{E}_{X_K|X_{-K},K}\left[f(X,K)\right]\cdot\mathbb{E}_{X_K|X_{-K},K}\left[g(X,K)\right]\big) \\
&\quad \text{(we can merge terms because the roles of } X \text{ and } Y \text{ are symmetric)} \\
&= \mathbb{E}_{K\sim p}\big[\mathbb{E}_{x_{-K}\sim p(x_{-K}|K)}\big[\mathbb{E}_{y_K\sim p(x_K|x_{-K},K)}\left[f(y,K)g(y,K)\right]-\mathbb{E}_{y_K\sim p(x_K|x_{-K},K)}\left[f(y,K)\right] \\
&\quad \cdot\mathbb{E}_{y_K\sim p(x_K|x_{-K},K)}\left[g(y,K)\right]\big]\big] \\
&= \mathbb{E}_{(X_{-K},K)\sim p_{\mathcal{X},\mathcal{K}}}\left[\mathrm{Cov}_{X_K|(X_{-K},K)}(f,g)\right]
\end{aligned}
\tag{4.27}
$$

which completes the proof. □

4.3.3.4 Proof of Theorem 4.1.3: Asymptotic sample complexity for adaptively-weighted MPLE

Note that Theorem 4.1.3 generalizes Theorem 4.1.2. To reduce proof duplication, we only write the proof for the more general Theorem 4.1.3 here. Notational definition for $p_\theta(x_K|x_{-K},K)$ and other background info are in Section 4.1.2.3.

Theorem 4.1.3 (Asymptotic variance of adaptively-weighted MPLE under a Poincaré Inequality, generalization of Theorem 4.1.2)**.** *Suppose the distribution p_{θ^*} satisfies a Poincaré inequality with constant C with respect to the adaptively-weighted block dynamics. Then under Assumption 4.1.1 and Assumption 4.1.2 where $p_\theta(x_K|x_{-K})$ is replaced by $p_\theta(x_K|x_{-K},K)$, the asymptotic variance of the adaptively-weighted MPLE can be bounded as: $\Gamma_{PL}\preceq C\mathcal{I}^{-1}$ where $\mathcal{I}$ is the Fisher Information matrix (Definition 4.1.1).*

Proof. By Lemma 4.3.3 and Lemma 4.3.9, for n training samples as $n\to\infty$, we have:

$$\sqrt{n}(\hat{\theta}_{PL}-\theta^*)\to\mathcal{N}\left(0,(\mathrm{Cov}_{(X,K)\sim p_{\mathcal{X},\mathcal{K}}}(-\nabla_\theta\log p_\theta(X_K|X_{-K},K))_{|\theta=\theta^*})^{-1}\right) \tag{4.28}$$

Now we relate the RHS of (4.28) to $\mathcal{I}$. Let d_Θ denote the dimensionality of θ, i.e. $\theta\in\mathbb{R}^{d_\Theta}$. Then, for any test vector $v\in\mathbb{R}^{d_\Theta}$ we have:

$$
\begin{aligned}
&v^\top\mathbb{E}_{(X,K)}\nabla_\theta\log p_\theta(x_K|x_{-S},K)\nabla_\theta\log p_\theta(x_K|x_{-S},K)^\top v_{|\theta=\theta^*} \\
&= \mathbb{E}_{(X,K)}(\nabla_\theta\log p_\theta(x_K|x_{-K},K)^\top v)^2_{|\theta=\theta^*} \\
&= \mathbb{E}_K\mathbb{E}_{x_{-K}|K}\mathrm{Var}_{x_K|x_{-K},K}(\nabla_\theta\log p_\theta(x_K|x_{-K},K)^\top v)_{|\theta=\theta^*}+(\mathbb{E}_{x_K|x_{-K},K}\nabla_\theta\log p_\theta(x_K|x_{-K},K)^\top v)^2_{|\theta=\theta^*}
\end{aligned}
\tag{4.29}
$$

Denote $f(X,K)\coloneqq\nabla_\theta\log p_\theta(x_K|x_{-K},K)^\top v$ Consider the two parts in Equation (4.29) separately: the first term is simply $\mathbb{E}_K\mathbb{E}_{x_{-K}|K}\mathrm{Var}_{x_S|x_{-K},K}(f(X,K))$, which, by Proposition 4.1.1, is equal to $\mathcal{E}_P(f,f)$. Moreover, by Poincaré inequality (Definition 4.1.5), this is $\geq\frac{1}{C}\mathrm{Var}_p(f)$.

The second term simplifies to

$$
\begin{aligned}
&\mathbb{E}_K\mathbb{E}_{x_{-K}|K}\left(\mathbb{E}_{x_K|x_{-K},K}\left[\nabla_\theta \log p_\theta(x_K|x_{-K},K)^\top v\right]\right)^2_{|\theta=\theta^*}\\
&= \mathbb{E}_K\mathbb{E}_{x_{-K}|K}\left(\mathbb{E}_{x_K|x_{-K},K}\left[\left(\frac{\nabla_\theta p_\theta(x_K|x_{-K},K)}{p_\theta(x_K|x_{-K},K)}\right)^\top v\right]\right)^2_{|\theta=\theta^*}\\
&= \mathbb{E}_K\mathbb{E}_{x_{-K}|K}\left[\int\left(\frac{\nabla_\theta p_\theta(x_K|x_{-K},K)}{p_\theta(x_K|x_{-K},K)}\right)^\top v\cdot p_{\mathcal{X},\mathcal{K}}(x_K|x_{-K},K)\,dx_K\right]^2_{|\theta=\theta^*}\\
&= \mathbb{E}_K\mathbb{E}_{x_{-K}|K}\left[\int\left(\nabla_\theta p_\theta(x_K|x_{-K},K)\right)^\top v\,dx_K\right]^2_{|\theta=\theta^*} \quad \text{, since } p_{\theta^*}=p_{\mathcal{X}} \text{ by Assumption 4.1.2 and Definition (4.1)}\\
&= \mathbb{E}_K\mathbb{E}_{x_{-K}|K}\left[\nabla_\theta\left(\int p_\theta(x_K|x_{-K},K)\,dx_K\right)^\top v\right]^2_{|\theta=\theta^*} \quad \text{(since } p_\theta \text{ is differentiable wrt } \theta \text{ by Assumption 4.1.1)}\\
&= 0
\end{aligned}
$$

Therefore, we have $\mathrm{Cov}_{(X,K)\sim p_{\mathcal{X},\mathcal{K}}}(-\nabla_\theta \log p_\theta(X_K|X_{-K},K))_{|\theta=\theta^*} \succeq \frac{1}{C}\mathcal{I}$.

Plugging into Equation (4.28), and using the monotonicity of the matrix inverse (Toda, 2011), we obtain the upper bound on the asymptotic variance of our estimator we want:

$$\Gamma_{PL} \preceq C\mathcal{I}^{-1}$$

□

4.3.4 Proof of Theorem 4.1.4: Generalization bound for learning the joint distribution

We first state our overall structure of the proof of Theorem 4.1.4, and then state and prove the key lemmas mentioned therein.

Theorem 4.1.4 (Generalization bound for learning the joint distribution)**.** *Let* $\hat{\theta} \coloneqq \arg\min_\theta \hat{L}_{PL}(\theta)$*. Under Assumption 4.1.3 and Assumption 4.1.4,* $\forall \epsilon > 0$*,* $\forall \delta \in (0,1)$*, with probability at least* $1-\delta$ *we have* $D_{\mathrm{TV}}\left(p_{\hat{\theta}}, p_{\mathcal{X}}\right) < \sqrt{\frac{1}{2}\bar{C}_{AT}(p_{\hat{\theta}})\left(\hat{L}_{PL}(\hat{\theta}) + B \cdot \ln\frac{1}{\beta} + \epsilon\right)} + C$ *where* $B = \sqrt{\frac{2^{3N}|\Omega|^N C_\epsilon(\Theta)}{m \cdot \delta}} + \sqrt{\frac{\ln\frac{8C_\epsilon(\Theta)}{\delta}}{2n}}$*, and* $C = \sqrt{\frac{|\Omega|^{3N}}{8\delta n}}$.

Proof. We first introduce a few pieces of notation. We will denote the data samples as $\mathcal{S}_{\mathcal{X}} \coloneqq \{X^{(i)} | X^{(i)} \sim p(X)\}$, $|\mathcal{S}_{\mathcal{X}}| = n$, and for each $X^{(i)}$ we sample m masks $\mathcal{S}_{\mathcal{K}}{}^{(i)} \coloneqq \{K_1^{(i)}, \ldots, K_m^{(i)}\}$ in which $K_j^{(i)}$ is sampled iid from $\mathcal{K}$ according to probabilities $p_{\mathcal{K}}(\cdot \mid X)$. [22] Theorem 4.1.4 follows by combining the following steps:

Step 1: relating closeness of the *conditional* distributions (i.e. the loss) to closeness of the *joint* distribution. The connection is established through the definition of the block-generalized approximate tensorization of entropy in Definition 4.1.9, by which we get[23]:

$$D_{\mathrm{KL}}\left(\tilde{p}_{\mathcal{X}}, p_{\hat{\theta}}\right) \le \bar{C}_{AT}(p_{\hat{\theta}})\tilde{L}_{PL}(\hat{\theta})$$

The details are in Proposition 4.3.1. By Pinsker's inequality, this implies

$$D_{\mathrm{TV}}\left(\tilde{p}_{\mathcal{X}}, p_{\hat{\theta}}\right) \le \sqrt{\frac{1}{2}D_{\mathrm{KL}}\left(\tilde{p}_{\mathcal{X}}, p_{\hat{\theta}}\right)} \le \sqrt{\frac{1}{2}\bar{C}_{AT}(p_{\hat{\theta}})\tilde{L}_{PL}(\hat{\theta})} \tag{4.30}$$

Step 2: generalization bound for learning the *conditional* distributions. We show that Assumption 4.1.3 and Assumption 4.1.4 imply a generalization guarantee for learning the *conditional* distributions from a finite sample of sequences and masked positions. We show that with probability at least $1 - \frac{\delta}{2}$, we have

$$\left|L_{PL}(\hat{\theta}) - \tilde{L}_{PL}(\hat{\theta})\right| < \left(\sqrt{\frac{2^{3N}|\Omega|^N C_\epsilon(\Theta)}{m \cdot \delta}} + \sqrt{\frac{\ln\frac{8C_\epsilon(\Theta)}{\delta}}{2n}}\right) \cdot \ln\frac{1}{\beta} + \epsilon \tag{4.31}$$

Proof details of this step are in Corollary 4.3.2.

Step 3: *empirical* joint distribution converges to *population* joint distribution. With probability at least $1 - \frac{\delta}{2}$, we have:

$$D_{\mathrm{TV}}\left(\tilde{p}_{\mathcal{X}}, p_{\mathcal{X}}\right) < \sqrt{\frac{|\Omega|^{3N}}{8\delta n}} \tag{4.32}$$

The proof of this is standard and details are in Lemma 4.3.6.

[22] Note that the $\{\cdot\}$ notation does not mean sets: duplicate entries are allowed in the training data $\mathcal{S}_{\mathcal{X}}$ and $\mathcal{S}_{\mathcal{K}}{}^{(i)}$.

[23] Recall, $\tilde{L}_{PL}$ is defined in (4.2)

Step 4: union bound and triangle inequality By union bound, with probability at least $1-\delta$, both Equation (4.31) and Equation (4.32) hold. Therefore, putting together the previous steps, we get:

$$D_{\mathrm{TV}}\left(p_{\hat{\theta}}, p_{\mathcal{X}}\right) \leq D_{\mathrm{TV}}\left(\tilde{p}_{\mathcal{X}}, p_{\hat{\theta}}\right) + D_{\mathrm{TV}}\left(\tilde{p}_{\mathcal{X}}, p_{\mathcal{X}}\right) \quad \text{(by triangle inequality)}$$

$$\leq \sqrt{\frac{1}{2}\bar{C}_{AT}(p_{\hat{\theta}})\tilde{L}_{\mathrm{PL}}(\hat{\theta})} + \sqrt{\frac{|\Omega|^{3N}}{8\delta n}} \quad \text{(by Equation (4.30) and Equation (4.32))}$$

$$< \sqrt{\frac{1}{2}\bar{C}_{AT}(p_{\hat{\theta}})\left(\hat{L}_{PL}(\hat{\theta}) + \left(\left(\sqrt{\frac{2^{3N}|\Omega|^{N} C_{\epsilon}(\Theta)}{m\cdot\delta}} + \sqrt{\frac{\ln\frac{8C_{\epsilon}(\Theta)}{\delta}}{2n}}\right)\cdot \ln\frac{1}{\beta} + \epsilon\right)\right)} + \sqrt{\frac{|\Omega|^{3N}}{8\delta n}}$$

(by Equation (4.31))

This completes the proof of the Theorem. □

We proceed to Step 1 first. We show:

Proposition 4.3.1. *$D_{\mathrm{KL}}\left(p_{\mathcal{X}}, p_{\theta}\right) \leq \bar{C}_{AT}(p_{\theta})L_{PL}(\theta)$ and $D_{\mathrm{KL}}\left(\tilde{p}_{\mathcal{X}}, p_{\theta}\right) \leq \bar{C}_{AT}(p_{\theta})\tilde{L}_{PL}(\theta)$*

Proof. By definition of block-generalized approximate tensorization of entropy in Definition 4.1.9

$$\begin{aligned} D_{\mathrm{KL}}(p_{\mathcal{X}}, p_{\theta}) &\leq \bar{C}_{AT}(p_{\theta}) \cdot \mathbb{E}_{X\sim p_{\mathcal{X}}}\left[\mathbb{E}_{K\sim p_{\mathcal{K}}(\cdot|X)}\left[D_{\mathrm{KL}}\left(p_{\mathcal{X}}(\cdot \mid X_{-K}, K), p_{\theta}(\cdot \mid X_{-K}, K)\right)\right]\right] \\ &= \bar{C}_{AT}(p_{\theta}) \cdot L_{\mathrm{PL}}(\theta) \end{aligned}$$

Likewise the latter holds when we replace p with $\tilde{p}$.

□

We will need the following simple observation in several concentration bounds we prove:

Proposition 4.3.2 (Bound on KL)**.** *Under Assumption 4.1.3,*

$$D_{\mathrm{KL}}\left(\tilde{p}_{\mathcal{X}}(\cdot|X_{-K}, K), p_{\theta}(\cdot|X_{-K}, K)\right) \in \left[0, \ln\frac{1}{\beta}\right]$$

Proof. By definition of D_{KL},

$$\begin{aligned} 0 \leq D_{\mathrm{KL}}\left(\tilde{p}_{\mathcal{X}}(\cdot|X_{-K}, K), p_{\theta}(\cdot|X_{-K}, K)\right) &= \sum_{X_K \in \Omega^{|K|}} \tilde{p}_{\mathcal{X}}(X_K|X_{-K}, K)\ln\frac{\tilde{p}_{\mathcal{X}}(X_K|X_{-K}, K)}{p_{\theta}(X_K|X_{-K}, K)} \\ &\leq \sum_{X_K \in \Omega^{|K|}} \tilde{p}_{\mathcal{X}}(X_K|X_{-K}, K)\ln\frac{1}{p_{\theta}(X_K|X_{-K}, K)} \\ &\leq \sum_{X_K \in \Omega^{|K|}} \tilde{p}_{\mathcal{X}}(X_K|X_{-K}, K)\ln\frac{1}{\beta} \quad \text{(by Assumption 4.1.3)} \\ &= \ln\frac{1}{\beta} \end{aligned}$$

□

we also recall a standard version of Hoeffding's inequality we'll use repeatedly:

Lemma 4.3.4 (Hoeffding's inequality)**.** *Let $Y_1, \cdots, Y_n$ be independent random variables such that $a \leq Y_i \leq b$ almost surely. Consider the sum of these random variables, $S_n = Y_1 + \cdots + Y_n$ whose expectation is $\mathbb{E}\left[S_n\right]$. Then, $\forall t > 0$, with probability at least $1 - 2e^{-\frac{2t^2}{n(b-a)^2}}$, we have $\left|S_n - \mathbb{E}\left[S_n\right]\right| < t$.*

Most of the generalization bounds we need for Step 2 (in particular, Corollary 4.3.2) will be derived from the following Lemma:

Lemma 4.3.5 (Point-wise generalization bound for learning conditional distributions)**.** *Fix a $\theta \in \Theta$ satisfying Assumption 4.1.3. $\forall \epsilon, t > 0$, with probability at least $1 - \frac{2^{N-2}|\Omega|^N}{\epsilon^2 m} - 2e^{-\frac{2t^2}{n\cdot\left(\ln\frac{1}{\beta}\right)^2}}$, we have*

$$\left|\hat{L}_{PL}(\theta) - \tilde{L}_{PL}(\theta)\right| < 2^N \epsilon \cdot \ln\frac{1}{\beta} + \frac{t}{n}$$

Proof. For notational convenience, let us denote by $\mathcal{S}_\mathcal{X}$ the training data points $\{X^{(i)}\}_{i\in[n]}$, and let us denote by $\mathcal{S}_\mathcal{K}(X)$ the set of masks corresponding to the training data point X.

Step 1: concentration over masked configurations

We first prove that $\hat{L}_{PL}(\theta)$ (Definition 4.1.2) concentrates to the expectation over masked positions K as m increases. [24]

Denote

$$f(X) \coloneqq \mathbb{E}_{K\sim p_\mathcal{K}(\cdot|X)}\left[D_{\mathrm{KL}}\left(\tilde{p}_\mathcal{X}(\cdot|X_{-K}, K), p_\theta(\cdot|X_{-K}, K)\right)\right] \tag{4.33}$$

Then the expectation of $\hat{L}_{PL}(\theta)$ over the randomness of $\mathcal{S}_\mathcal{K}$ is:

$$\begin{aligned}
\mathbb{E}_{\{\mathcal{S}_\mathcal{K}(j)\,|\,j\in[n]\}}\left[\hat{L}_{PL}(\theta)\right] &= \frac{1}{n}\sum_{X\in\mathcal{S}_\mathcal{X}} \frac{1}{m}\mathbb{E}_{\mathcal{S}_\mathcal{K}(j)}\left[\sum_{K\in\mathcal{S}_\mathcal{K}(X)} D_{\mathrm{KL}}\left(\tilde{p}(\cdot|X_{-K}, K), p_\theta(\cdot|X_{-K}, K)\right)\right] \\
&= \frac{1}{n}\sum_{X\in\mathcal{S}_\mathcal{X}} \mathbb{E}_{K\sim p_\mathcal{K}(\cdot|X)}\left[D_{\mathrm{KL}}\left(\tilde{p}_\mathcal{X}(\cdot|X_{-K}, K), p_\theta(\cdot|X_{-K}, K)\right)\right] \\
&= \frac{1}{n}\sum_{X\in\mathcal{S}_\mathcal{X}} f(X) \qquad (4.34)
\end{aligned}$$

Moreover, for each K, the (observed) empirical probability $p_S(K \mid X)$ converges to the true probability $p_\mathcal{K}(K \mid X)$ as m increases, because the count, $p_S(K \mid X)\cdot m$, follows the binomial distribution $\mathrm{Binomial}(m, p_\mathcal{K}(K \mid X))$. More specifically, by Chebyshev's inequality, $\forall\epsilon > 0$, and a fixed X we have:

$$\begin{aligned}
\mathbb{P}\left\{|p_S(K \mid X) - p_\mathcal{K}(K \mid X)| \ge \epsilon\right\} &= \mathbb{P}\left\{|p_S(K \mid X)m - p_\mathcal{K}(K \mid X)m| \ge \epsilon m\right\} \\
&\le \frac{\mathrm{Var}\left(p_S(K \mid X)m\right)}{\epsilon^2 m^2} \quad \text{(Chebyshev's inequality)} \\
&= \frac{m p_\mathcal{K}(K \mid X)(1 - p_\mathcal{K}(K \mid X))}{\epsilon^2 m^2} \quad \text{(since } p_S(K)m \sim \mathrm{Binomial}(m, p(K))\text{)} \\
&= \frac{p_\mathcal{K}(K \mid X)(1 - p_\mathcal{K}(K \mid X))}{\epsilon^2 m} \\
&\le \frac{1}{4\epsilon^2 m}
\end{aligned}$$

Applying union bound over $K \in \{0,1\}^N, X \in \mathcal{X}$,

$$\mathbb{P}\left\{|p_S(K \mid X) - p_\mathcal{K}(K \mid X)| < \epsilon, \forall K \in \{0,1\}^N, \forall X \in \mathcal{S}_\mathcal{X}\right\} \ge 1 - \frac{2^N|\Omega|^N}{4\epsilon^2 m} = 1 - \frac{2^{N-2}|\Omega|^N}{\epsilon^2 m} \tag{4.35}$$

[24] Note that the terms $D_{\mathrm{KL}}\left(\tilde{p}_\mathcal{X}(\cdot|X_{-K}, K), p_\theta(\cdot|X_{-K}, K)\right)$ are (generally) not independent for different K. Besides, the terms $\sum_{K\in\mathcal{S}_\mathcal{K}} D_{\mathrm{KL}}\left(\tilde{p}_\mathcal{X}(\cdot|X_{-K}, K), p_\theta(\cdot|X_{-K}, K)\right)$ are (generally) not independent for different $\mathcal{S}_\mathcal{K}$.

Plugging into Equation (4.33) and Equation (4.34), we get with probability at least $1 - \frac{2^{N-2}|\Omega|^N}{\epsilon^2 m}$,

$$\begin{aligned}
&\left|\hat{L}_{PL}(\theta) - \frac{1}{n}\sum_{X \in \mathcal{S}_\mathcal{X}} f(X)\right| \\
&= |\frac{1}{n}\sum_{X \in \mathcal{S}_\mathcal{X}} \frac{1}{|\mathcal{S}_\mathcal{K}(X)|}\sum_{K \in \mathcal{S}_\mathcal{K}(X)} D_{\mathrm{KL}}\left(\tilde{p}_\mathcal{X}(\cdot|X_{-K}, K), p_\theta(\cdot|X_{-K}, K)\right) \\
&\quad - \frac{1}{n}\sum_{X \in \mathcal{S}_\mathcal{X}} \mathbb{E}_K\left[D_{\mathrm{KL}}\left(\tilde{p}_\mathcal{X}(\cdot|X_{-K}, K), p_\theta(\cdot|X_{-K}, K)\right)\right]| \\
&\leq \frac{1}{n}\sum_{X \in \mathcal{S}_\mathcal{X}}\sum_{K \in \{0,1\}^N} |p_S(K \mid X) - p_\mathcal{K}(K \mid X)| \cdot D_{\mathrm{KL}}\left(\tilde{p}_\mathcal{X}(\cdot|X_{-K}, K), p_\theta(\cdot|X_{-K}, K)\right) \quad \text{(triangle inequality)} \\
&< \frac{1}{n}\sum_{X \in \mathcal{S}_\mathcal{X}}\sum_{K \in \{0,1\}^N} \epsilon \cdot D_{\mathrm{KL}}\left(\tilde{p}_\mathcal{X}(\cdot|X_{-K}, K), p_\theta(\cdot|X_{-K}, K)\right) \quad \text{(by Equation (4.35))} \\
&\leq \frac{1}{n}\sum_{X \in \mathcal{S}_\mathcal{X}}\sum_{K \in \{0,1\}^N} \epsilon \cdot \ln\frac{1}{\beta} \quad \text{(by Proposition 4.3.2)} \\
&= \frac{1}{n}\sum_{X \in \mathcal{S}_\mathcal{X}} 2^N \epsilon \cdot \ln\frac{1}{\beta} \quad = 2^N \epsilon \cdot \ln\frac{1}{\beta}
\end{aligned} \tag{4.36}$$

Step 2: concentration over sequences X in training data.
Recall $f(X)$ defined in Equation (4.33). We have:

$$\begin{aligned}
\mathbb{E}\left[f(X)\right] &= \mathbb{E}_{X \sim \tilde{p}_\mathcal{X}}\left[\mathbb{E}_{K \sim p_\mathcal{K}(\cdot|X)}\left[D_{\mathrm{KL}}\left(\tilde{p}(\cdot|X_{-K}, K), p_\theta(\cdot|X_{-K}, K)\right)\right]\right] \\
&= \tilde{L}_{\mathrm{PL}}(\theta)
\end{aligned}$$

Note that $f(X) \in [0, \ln\frac{1}{\beta}]$ by Proposition 4.3.2. Thus, applying Hoeffding's inequality (Lemma 4.3.4), $\forall t > 0$, with probability at least $1 - 2e^{-\frac{2t^2}{n \cdot \left(\ln\frac{1}{\beta}\right)^2}}$, we have

$$\left|\frac{1}{n}\sum_{X \in \mathcal{S}_\mathcal{X}} f(X) - \mathbb{E}\left[f(X)\right]\right| < \frac{t}{n} \tag{4.37}$$

Step 3: combining results: concentration over both masks K and sequences X.
By union bound, with probability at least

$$1 - \frac{2^{N-2}\left|\Omega\right|^N}{\epsilon^2 m} - 2e^{-\frac{2t^2}{n \cdot \left(\ln\frac{1}{\beta}\right)^2}}$$

both Equation (4.36) and Equation (4.37) hold, giving us:

$$\begin{aligned}
&\left|\hat{L}_{PL}(\theta) - \tilde{L}_{\mathrm{PL}}(\theta)\right| \\
&\leq \left|\hat{L}_{PL}(\theta) - \frac{1}{n}\sum_{X \in \mathcal{S}_\mathcal{X}} f(X)\right| + \left|\frac{1}{n}\sum_{X \in \mathcal{S}_\mathcal{X}} f(X) - \tilde{L}_{\mathrm{PL}}(\theta)\right| \quad \text{(triangle inequality)} \\
&< 2^N \epsilon \cdot \ln\frac{1}{\beta} + \frac{t}{n}
\end{aligned}$$

□

Remark 4.3.1. *The two terms in the bound given by Lemma 4.3.5, i.e.* $2^N\epsilon \cdot \ln\frac{1}{\beta}$ *and* $\frac{t}{n}$*, can be controlled by setting appropriate* ϵ *and* t *based on* m *and* n*, respectively. These two terms can reduce by increasing* m *and* n*, respectively, as we will show in the subsequent corollary. This is intuitive: we expect a smaller generalization gap when the model is trained on more mask configurations for each sequence, and when more sequences are included in the data. The first term grows with* N *— this is also intuitive: when the sequences are longer, it is natural to require observing more mask configurations.*

Corollary 4.3.1 (Point-wise generalization bound for learning conditional distributions, special case)**.** *Fix a* $\theta \in \Theta$ *satisfying Assumption 4.1.3. with probability at least* $1-\delta$*, we have*

$$\left|\hat{L}_{PL}(\theta) - \tilde{L}_{PL}(\theta)\right| < \left(\sqrt{\frac{2^{3N-1}\left|\Omega\right|^N}{m\cdot\delta}} + \sqrt{\frac{\ln\frac{4}{\delta}}{2n}}\right)\cdot \ln\frac{1}{\beta}$$

Proof. Apply Lemma 4.3.5 with ϵ and t satisfying

$$\epsilon = \sqrt{\frac{2^{N-1}\left|\Omega\right|^N}{m\cdot\delta}}$$
$$t = \sqrt{\frac{\ln\frac{4}{\delta}\cdot n}{2}}\ln\frac{1}{\beta}$$

we have that with probability at least $1-\delta$, it holds:

$$\begin{aligned}&\left|\hat{L}_{PL}(\theta) - \tilde{L}_{\mathrm{PL}}(\theta)\right| \\ &< 2^N\epsilon\cdot\ln\frac{1}{\beta} + \frac{t}{n} \\ &= \left(\sqrt{\frac{2^{3N-1}\left|\Omega\right|^N}{m\cdot\delta}} + \sqrt{\frac{\ln\frac{4}{\delta}}{2n}}\right)\cdot\ln\frac{1}{\beta}\end{aligned}$$

□

Corollary 4.3.2 (Uniform convergence generalization bound for learning conditional distributions)**.** *Under Assumption 4.1.3 and Assumption 4.1.4,* $\forall\delta\in(0,1)$*,* $\forall\epsilon>0$*, with probability at least* $1-\delta$*, we have*

$$\left|\hat{L}_{PL}(\theta) - \tilde{L}_{PL}(\theta)\right| < \left(\sqrt{\frac{2^{3N-1}\left|\Omega\right|^N C_\epsilon(\Theta)}{m\cdot\delta}} + \sqrt{\frac{\ln\frac{4C_\epsilon(\Theta)}{\delta}}{2n}}\right)\cdot\ln\frac{1}{\beta} + \epsilon$$

Proof. By Assumption 4.1.4, let $C_\epsilon(\Theta)$ denote the complexity of parameter space Θ, with the corresponding partition $\mathrm{Par}_\epsilon(\Theta) = \{\Theta_1, \cdots, \Theta_{C_\epsilon(\Theta)}\}$. As a corollary of Assumption 4.1.4, $\forall i, \forall\theta_1,\theta_2\in\Theta_i$, $\left|\tilde{L}_{PL}(\theta_1) - \tilde{L}_{PL}(\theta_2)\right| \le \frac{\epsilon}{2}$, $\left|\hat{L}_{PL}(\theta_1) - \hat{L}_{PL}(\theta_2)\right| \le \frac{\epsilon}{2}$.

Moreover, for each $i\in[C_\epsilon(\Theta)]$, arbitrarily select any point $\theta_i^*\in\Theta_i$ (as a "representative" of that region of the parameter space). Let the set of "representative points" be $\Theta^* = \{\theta_i^* \mid i\in[C_\epsilon(\Theta)]\}$. By Corollary 4.3.1, fixing any $\theta\in\Theta$ satisfying Assumption 4.1.3, then with probability at least $1-\frac{\delta}{C_\epsilon(\Theta)}$, we have

$$\left|\hat{L}_{PL}(\theta) - \tilde{L}_{\mathrm{PL}}(\theta)\right| < \left(\sqrt{\frac{2^{3N-1}\left|\Omega\right|^N C_\epsilon(\Theta)}{m\cdot\delta}} + \sqrt{\frac{\ln\frac{4C_\epsilon(\Theta)}{\delta}}{2n}}\right)\cdot\ln\frac{1}{\beta}$$

Applying union bound over $\theta_i^* \in \Theta^*$, since $|\Theta^*| = C_\epsilon(\Theta)$, with probability at least $1-\delta$,

$$\forall i \in [C_\epsilon(\Theta)], \quad \left|L_{\mathrm{PL}}(\theta_i^*) - \tilde{L}_{\mathrm{PL}}(\theta_i^*)\right| < \left(\sqrt{\frac{2^{3N-1}\left|\Omega\right|^N C_\epsilon(\Theta)}{m \cdot \delta}} + \sqrt{\frac{\ln \frac{4C_\epsilon(\Theta)}{\delta}}{2n}}\right) \cdot \ln \frac{1}{\beta} \tag{4.38}$$

Finally, by Assumption 4.1.4, $\forall \theta \in \Theta$, there exists $i \in [C_\epsilon(\Theta)]$ such that $\theta \in \Theta_i$ (i.e. θ falls into that partition), and

$$\begin{aligned}\left|\tilde{L}_{\mathrm{PL}}(\theta) - \tilde{L}_{\mathrm{PL}}(\theta_i^*)\right| &\leq \frac{\epsilon}{2} \\ \left|\hat{L}_{PL}(\theta) - \hat{L}_{PL}(\theta_i^*)\right| &\leq \frac{\epsilon}{2}\end{aligned} \tag{4.39}$$

Combining Equation (4.38) and Equation (4.39) gives

$$\begin{aligned}&\left|\hat{L}_{PL}(\theta) - \tilde{L}_{\mathrm{PL}}(\theta)\right| \\ &\leq \left|\hat{L}_{PL}(\theta) - \hat{L}_{PL}(\theta_i^*)\right| + \left|\hat{L}_{PL}(\theta_i^*) - \tilde{L}_{\mathrm{PL}}(\theta_i^*)\right| \\ &\quad + \left|\tilde{L}_{\mathrm{PL}}(\theta_i^*) - \tilde{L}_{\mathrm{PL}}(\theta)\right| \quad \text{(by triangle inequality)} \\ &< \frac{\epsilon}{2} + \left(\sqrt{\frac{2^{3N-1}\left|\Omega\right|^N C_\epsilon(\Theta)}{m \cdot \delta}} + \sqrt{\frac{\ln \frac{4C_\epsilon(\Theta)}{\delta}}{2n}}\right) \cdot \ln \frac{1}{\beta} + \frac{\epsilon}{2} \quad \text{(by Equation (4.38) and Equation (4.39))} \\ &= \left(\sqrt{\frac{2^{3N-1}\left|\Omega\right|^N C_\epsilon(\Theta)}{m \cdot \delta}} + \sqrt{\frac{\ln \frac{4C_\epsilon(\Theta)}{\delta}}{2n}}\right) \cdot \ln \frac{1}{\beta} + \epsilon\end{aligned}$$

□

Finally, we complete Step 3 (proving Equation (4.32)):

Lemma 4.3.6 (Empirical PMF converges to population PMF)**.** *For any $\delta > 0$, with probability at least $1-\delta$, we have:*

$$D_{\mathrm{TV}}\left(\tilde{p}_{\mathcal{X}}, p_{\mathcal{X}}\right) < \sqrt{\frac{\left|\Omega\right|^{3N}}{16\delta n}}$$

Proof. $\forall X \in \Omega^N$, the number of times that X appears in the training data $\mathcal{S}_{\mathcal{X}}$ follows the binomial distribution

$$\tilde{p}_{\mathcal{X}}(X)n \sim \mathrm{Binomial}(n, p_{\mathcal{X}}(X))$$

with mean $np_{\mathcal{X}}(X)$ and variance $np_{\mathcal{X}}(X)(1-p_{\mathcal{X}}(X))$. Hence, by Chebyshev's inequality, $\forall \epsilon > 0$

$$\begin{aligned}\mathbb{P}\left\{\left|\tilde{p}_{\mathcal{X}}(X) - p_{\mathcal{X}}(X)\right| \geq \epsilon\right\} &= \mathbb{P}\left\{\tilde{p}_{\mathcal{X}}(X)n - p_{\mathcal{X}}(X)n \geq \epsilon n\right\} \\ &\leq \frac{\mathrm{Var}\left(\tilde{p}_{\mathcal{X}}(X)n\right)}{\epsilon^2 n^2} \quad \text{(Chebyshev's inequality)} \\ &= \frac{np_{\mathcal{X}}(X)(1-p_{\mathcal{X}}(X))}{\epsilon^2 n^2} \quad \text{(since } \tilde{p}_{\mathcal{X}}(X)n \sim \mathrm{Binomial}(n, p_{\mathcal{X}}(X))\text{)} \\ &= \frac{p_{\mathcal{X}}(X)(1-p_{\mathcal{X}}(X))}{\epsilon^2 n} \\ &\leq \frac{1}{4\epsilon^2 m}\end{aligned}$$

Applying union bound over $X \in \Omega^N$,

$$\mathbb{P}\left\{|\tilde{p}_{\mathcal{X}}(X) - p_{\mathcal{X}}(X)| < \epsilon, \forall X \in \Omega^N\right\} \geq 1 - \frac{|\Omega|^N}{4\epsilon^2 n} \tag{4.40}$$

Hence, we get with probability at least $1 - \frac{|\Omega|^N}{4\epsilon^2 n}$,

$$D_{\mathrm{TV}}\left(\tilde{p}_{\mathcal{X}}, p_{\mathcal{X}}\right) = \frac{1}{2} \sum_{X \in \Omega^N} |\tilde{p}_{\mathcal{X}}(X) - p_{\mathcal{X}}(X)| < \frac{1}{2} \sum_{X \in \Omega^N} \epsilon = \frac{1}{2} |\Omega|^N \epsilon \tag{4.41}$$

Solving for $\delta = \frac{|\Omega|^N}{4\epsilon^2 n}$ gives $\epsilon = \sqrt{\frac{|\Omega|^N}{4\delta n}}$. Therefore, by Equation (4.40), with probability at least $1 - \delta$, we have

$$D_{\mathrm{TV}}\left(\tilde{p}_{\mathcal{X}}, p_{\mathcal{X}}\right) < \frac{1}{2} |\Omega|^N \epsilon = \sqrt{\frac{|\Omega|^{3N}}{16\delta n}}$$

□

4.3.5 Proof of Proposition 4.3.3: Modes of the strongly ferromagnetic Ising model

This section provides formal proofs for the discussion under Assumption 4.1.5 in Section 4.1.4.2.

Proposition 4.3.3 (Modes of the strongly ferromagnetic Ising model)**.** *On Ising model G in Equation* (4.6) *under Assumption 4.1.5, the regions $\mathcal{R}_1$ and $\mathcal{R}_{-1}$ defined in Equation* (4.7) *and Equation* (4.8) *satisfy:*

1. *$\forall \boldsymbol{x} \in \mathcal{R}_1$, $\forall \boldsymbol{y} \in \mathcal{R}_{-1}$, $\forall \boldsymbol{z} \in \{-1, 1\}^N \setminus (\mathcal{R}_1 \cup \mathcal{R}_{-1})$: $p_G(\boldsymbol{x}) > p_G(\boldsymbol{y}) > e^{2J_0} p_G(\boldsymbol{z})$*
2. *There exists a bijection $f : \mathcal{R}_1 \mapsto \mathcal{R}_{-1}$ such that $\forall \boldsymbol{x} \in \mathcal{R}_1$, $p_G(\boldsymbol{x}) = e^{2h_G} p_G(f(\boldsymbol{x}))$*

Proof. Consider $\boldsymbol{x} \in \mathcal{R}_1, \boldsymbol{y} \in \mathcal{R}_{-1}$. We have:

$$\begin{aligned}
\frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{y})} &= \frac{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i \mathbf{x}_i + \sum_{i\neq j\in C_G\subset[N]} J\mathbf{x}_i\mathbf{x}_j\right)}{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i \mathbf{y}_i + \sum_{i\neq j\in C_G\subset[N]} J\mathbf{y}_i\mathbf{y}_j\right)} \\
&= \frac{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i \mathbf{x}_i\right)}{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i \mathbf{y}_i\right)} \quad (\text{since } \mathbf{x}_i\mathbf{x}_j = \mathbf{y}_i\mathbf{y}_j = 1,\ \forall \boldsymbol{x} \in \mathcal{R}_1, \forall \boldsymbol{y} \in \mathcal{R}_{-1}) \\
&= \frac{\exp\left(\sum_{i\in C_G} \boldsymbol{h}_i \mathbf{x}_i + \sum_{i\notin C_G} \boldsymbol{h}_i \mathbf{x}_i\right)}{\exp\left(\sum_{i\in C_G} \boldsymbol{h}_i \mathbf{y}_i + \sum_{i\notin C_G} \boldsymbol{h}_i \mathbf{y}_i\right)} \\
&= \frac{\exp\left(\sum_{i\in C_G} \boldsymbol{h}_i + \sum_{i\notin C_G} \boldsymbol{h}_i \mathbf{x}_i\right)}{\exp\left(-\sum_{i\in C_G} \boldsymbol{h}_i + \sum_{i\notin C_G} \boldsymbol{h}_i \mathbf{y}_i\right)} \quad (\text{since } \boldsymbol{x} \in \mathcal{R}_1, \boldsymbol{y} \in \mathcal{R}_{-1}) \\
&\geq \frac{\exp\left(\sum_{i\in C_G} \boldsymbol{h}_i - \sum_{i\notin C_G} |\boldsymbol{h}_i|\right)}{\exp\left(-\sum_{i\in C_G} \boldsymbol{h}_i + \sum_{i\notin C_G} |\boldsymbol{h}_i|\right)} \quad (\text{since } \mathbf{x}_i, \mathbf{y}_i \in \pm 1) \\
&= \exp\left(2\sum_{i\in C_G} \boldsymbol{h}_i - 2\sum_{i\notin C_G} |\boldsymbol{h}_i|\right) \\
&> \exp(0) \quad (\text{by Assumption 4.1.5}) \\
&= 1
\end{aligned}$$

On the other hand, if we consider $\boldsymbol{y} \in \mathcal{R}_{-1}, \boldsymbol{z} \in \{-1, 1\}^N \setminus (\mathcal{R}_1 \cup \mathcal{R}_{-1})$, we have:

$$\begin{aligned}
\frac{p_G(\boldsymbol{y})}{p_G(\boldsymbol{z})} &= \frac{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{y}_i + \sum_{i\neq j\in C_G\subset[N]} J\mathbf{y}_i\mathbf{y}_j\right)}{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{z}_i + \sum_{i\neq j\in C_G\subset[N]} J\mathbf{z}_i\mathbf{z}_j\right)} \\
&= \frac{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{y}_i + \sum_{i\neq j\in C_G\subset[N]} J\right)}{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{z}_i + \sum_{i\neq j\in C_G\subset[N]} J\mathbf{z}_i\mathbf{z}_j\right)} \quad (\text{since } \mathbf{y}_i\mathbf{y}_j = 1,\, \forall \boldsymbol{y} \in \mathcal{R}_{-1}) \\
&\geq \frac{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{y}_i + \sum_{i\neq j\in C_G\subset[N]} J\right)}{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{z}_i + \sum_{i\neq j\in C_G\subset[N]} J - 2(|C_G|-1)J\right)}
\end{aligned}$$

the above is because, since $\boldsymbol{z} \in \{-1,1\}^N \setminus (\mathcal{R}_1 \cup \mathcal{R}_{-1})$, obviously the denominator

$$\sum_{i\neq j\in C_G\subset[N]} J\mathbf{z}_i\mathbf{z}_j \leq \sum_{i\neq j\in C_G\subset[N]} J - 2(|C_G|-1)J$$

$$\begin{aligned}
&= \frac{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{y}_i\right)}{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{z}_i - 2(|C_G|-1)J\right)} \geq \frac{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{y}_i\right)}{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{z}_i - 2J\right)} \geq \frac{\exp\left(-\|\boldsymbol{h}\|_1\right)}{\exp\left(\|\boldsymbol{h}\|_1 - 2J\right)} \\
&= \exp\left(2(J - \|\boldsymbol{h}\|_1)\right) \geq \exp\left(2J_0\right) \quad (\text{by Assumption 4.1.5})
\end{aligned}$$

Proceeding to Part 2, let's define $f : \mathcal{R}_1 \mapsto \mathcal{R}_{-1}$ as:

$$\forall \boldsymbol{x} \in \mathcal{R}_1, \quad f(\boldsymbol{x})_i = \begin{cases} -1, & \text{if } i \in C_G \\ \boldsymbol{x}_i, & \text{if } i \notin C_G \end{cases}$$

Let $\boldsymbol{w} \coloneqq f(\boldsymbol{x})$. Then,

$$\begin{aligned}
\frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{w})} &= \frac{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{x}_i + \sum_{i\neq j\in C_G\subset[N]} J\mathbf{x}_i\mathbf{x}_j\right)}{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{w}_i + \sum_{i\neq j\in C_G\subset[N]} J\mathbf{w}_i\mathbf{w}_j\right)} \\
&= \frac{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{x}_i\right)}{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i\mathbf{w}_i\right)} \quad (\text{since } \mathbf{x}_i\mathbf{x}_j = \mathbf{w}_i\mathbf{w}_j = 1,\, \forall \boldsymbol{x} \in \mathcal{R}_1, \forall \boldsymbol{w} \in \mathcal{R}_{-1}) \\
&= \frac{\exp\left(\sum_{i\in C_G} \boldsymbol{h}_i\mathbf{x}_i + \sum_{i\notin C_G} \boldsymbol{h}_i\mathbf{x}_i\right)}{\exp\left(\sum_{i\in C_G} \boldsymbol{h}_i\mathbf{w}_i + \sum_{i\notin C_G} \boldsymbol{h}_i\mathbf{w}_i\right)} \\
&= \frac{\exp\left(\sum_{i\in C_G} \boldsymbol{h}_i + \sum_{i\notin C_G} \boldsymbol{h}_i\mathbf{x}_i\right)}{\exp\left(-\sum_{i\in C_G} \boldsymbol{h}_i + \sum_{i\notin C_G} \boldsymbol{h}_i\mathbf{w}_i\right)} \quad (\text{since } \boldsymbol{x} \in \mathcal{R}_1, \boldsymbol{w} \in \mathcal{R}_{-1}) \\
&= \frac{\exp\left(\sum_{i\in C_G} \boldsymbol{h}_i\right)}{\exp\left(-\sum_{i\in C_G} \boldsymbol{h}_i\right)} \quad (\text{since } \boldsymbol{w}_i = \boldsymbol{x}_i,\, \forall i \notin C_G) \\
&= \exp\left(2\sum_{i\in C_G} \boldsymbol{h}_i\right) \\
&= \exp\left(2h_G\right) \quad (\text{by Assumption 4.1.5})
\end{aligned}$$

□

4.3.6 Proof of Proposition 4.1.4: k-Gibbs sampler can reach the mode fast

Proposition 4.1.4 (k-Gibbs sampler can reach the mode fast)**.** *Consider the Ising model in Equation (4.6) satisfying Assumption 4.1.5. Let us denote* $(0,1) \ni c_{\mathcal{R}_1} \coloneqq 1 - \frac{\binom{N-|C_G|}{k-|C_G|}}{\binom{N}{k}} \frac{e^{2(J_0+h_G)}}{e^{2(J_0+h_G)}+e^{2J_0}+2^{|C_G|}-2}$.

Then, for any initial $\boldsymbol{X}^{(0)}$ *and* $\delta \in (0,1)$*, with probability at least* $1-\delta$*, after* $T \coloneqq \left\lceil \log_{c_{\mathcal{R}_1}} \delta \right\rceil$ *steps of* k-***Gibbs sampler*** *(Definition 4.1.10) with* $k \geq |C_G|$*, we have* $\{\boldsymbol{X}^{(t)} | t \in [T]\} \cap \mathcal{R}_1 \neq \emptyset$*.*

Proof. At any step, let K (with $|K| = k$) denote the set of coordinates to re-sample. We first consider the probability of $C_G \subset K$, which allows the whole C_G to be updated jointly: The total number of ways to select K is $\binom{N}{k}$. Now count the number of K's which satisfy $C_G \subset K$. This is the same as selecting the remaining $k - |C_G|$ coordinates from the other $N - |C_G|$ coordinates which are not in C_G. So there are $\binom{N-|C_G|}{k-|C_G|}$ distinct K's which satisfy $C_G \subset K$.

$$\mathbb{P}\{C_G \subset K\} = \frac{\binom{N-|C_G|}{k-|C_G|}}{\binom{N}{k}} \tag{4.42}$$

$\forall t \in \mathbb{N}, \forall \boldsymbol{X}^{(t)} \in \{-1,1\}^N$, and $K \in [N]$ such that $|K| = k$ and $C_G \subset K$, consider $X_K^{(t+1)} \sim p_G(\cdot \mid X_{-K}^{(t)})$. There are three cases (whhich exhaust all possibilities):

1. $\boldsymbol{X}^{(t+1)} \in \mathcal{R}_1$
2. $\boldsymbol{X}^{(t+1)} \in \mathcal{R}_{-1}$
3. $\boldsymbol{X}^{(t+1)} \in \{-1,1\}^N \backslash (\mathcal{R}_1 \cup \mathcal{R}_{-1})$

We will use Proposition 4.3.3 to show that Case 1 occurs with probability at least a constant. We have:

$$\mathbb{P}\left\{\boldsymbol{X}^{(t+1)} \in \mathcal{R}_1\right\} = e^{2h_G} \mathbb{P}\left\{\boldsymbol{X}^{(t+1)} \in \mathcal{R}_{-1}\right\}$$
$$\frac{\mathbb{P}\left\{\boldsymbol{X}^{(t+1)} \in \mathcal{R}_{-1}\right\}}{\mathbb{P}\left\{\boldsymbol{X}^{(t+1)} \in \{-1,1\}^N \backslash (\mathcal{R}_1 \cup \mathcal{R}_{-1})\right\}} \geq e^{2J_0} \frac{|\mathcal{R}_{-1}|}{|\{-1,1\}^N \backslash (\mathcal{R}_1 \cup \mathcal{R}_{-1})|} = \frac{e^{2J_0}}{2^{|C_G|}-2}$$

Since the probabilities of the three cases sum up to 1,

$$\mathbb{P}\left\{\boldsymbol{X}^{(t+1)} \in \mathcal{R}_1\right\} \geq \frac{e^{2(J_0+h_G)}}{e^{2(J_0+h_G)} + e^{2J_0} + 2^{|C_G|} - 2}$$

Combining with Equation (4.42), we have $\forall t \in \mathbb{N}$ and $\forall \boldsymbol{X}^{(t)} \in \{-1,1\}^N$, it holds that:

$$\mathbb{P}\left\{\boldsymbol{X}^{(t+1)} \in \mathcal{R}_1\right\} \geq \mathbb{P}\left\{C_G \subset K, \boldsymbol{X}^{(t+1)} \in \mathcal{R}_1\right\} \geq \frac{\binom{N-|C_G|}{k-|C_G|}}{\binom{N}{k}} \frac{e^{2(J_0+h_G)}}{e^{2(J_0+h_G)} + e^{2J_0} + 2^{|C_G|} - 2} \coloneqq 1 - c_{\mathcal{R}_1}$$

From this it follows that

$$\mathbb{P}\left\{\{\boldsymbol{X}^{(t)} | t \in [T]\} \cap \mathcal{R}_1 = \emptyset\right\} \leq c_{\mathcal{R}_1}^T$$

Therefore, when $T \geq \log_{c_{\mathcal{R}_1}} \delta$,

$$\mathbb{P}\left\{\{\boldsymbol{X}^{(t)} | t \in [T]\} \cap \mathcal{R}_1 = \emptyset\right\} \leq \delta$$

□

4.3.7 Proof of Proposition 4.1.5 independent parallel sampling stuck in bad samples

Proposition 4.1.5 (Independent parallel sampling stuck in bad samples)**.** *Consider the Ising model in Equation (4.6) satisfying Assumption 4.1.5. Let us denote* $c_{stuck} \coloneqq \frac{2\left(-1+\frac{1-\exp(-2J_0)}{\exp(-2J_0)+1}\frac{|C_G|}{2}\right)^2}{|C_G|}$.

For an initial $\boldsymbol{X}^{(0)}$ *such that* $\sum_{i \in C_G} \boldsymbol{X}_i^{(0)} \le -2$, *for any* $\delta \in (0,1)$, *with probability at least* $1-\delta$, *after* $T \coloneqq \left\lfloor \frac{\delta}{2} \exp(c_{stuck}) \right\rfloor$ *steps of* ***independent parallel*** *(Definition 4.1.11), we have* $\forall t \in [T], \sum_{i \in C_G} \boldsymbol{X}_i^{(t)} \le -2$.

Proof. Suppose at step t, $\boldsymbol{X}^{(t)}$ is such that $\sum_{i \in C_G} \boldsymbol{X}_i^{(t)} \le -2$ (satisfied at $t = 0$), then

$$\forall j \in C_G, \sum_{i \in C_G, i \neq j} \boldsymbol{X}_i^{(t)} \le -1 \tag{4.43}$$

Hence its next-step distribution $X_j^{(t+1)} \sim p(\cdot \mid X_{-\{j\}}^{(t)})$ satisfies

$$\begin{aligned}
\frac{\mathbb{P}\left\{X_j^{(t+1)} = 1\right\}}{\mathbb{P}\left\{X_j^{(t+1)} = -1\right\}} &= \frac{\exp\left(\sum_{i \in [N]} \boldsymbol{h}_i \mathbf{x}_i + \sum_{i \neq j \in C_G \subset [N]} J \mathbf{x}_i \mathbf{x}_j\right)|_{\mathbf{x}_j = 1}}{\exp\left(\sum_{i \in [N]} \boldsymbol{h}_i \mathbf{x}_i + \sum_{i \neq j \in C_G \subset [N]} J \mathbf{x}_i \mathbf{x}_j\right)|_{\mathbf{x}_j = -1}} \quad \text{(by definition in Equation (4.6))} \\
&= \frac{\exp\left(\boldsymbol{h}_j + \sum_{i \in C_G, i \neq j} J \mathbf{x}_i\right)}{\exp\left(-\boldsymbol{h}_j - \sum_{i \in C_G, i \neq j} J \mathbf{x}_i\right)} \quad \text{(canceling the same terms)} \\
&= \exp\left(2\boldsymbol{h}_j + 2J \sum_{i \in C_G, i \neq j} \mathbf{x}_i\right) \\
&\le \exp(2\boldsymbol{h}_j - 2J) \quad \text{(by Equation (4.43))} \\
&\le \exp(-2J_0) \quad \text{(by Assumption 4.1.5)}
\end{aligned}$$

Therefore

$$X_j^{(t+1)} = \begin{cases} 1, & \text{with prob } \le \frac{\exp(-2J_0)}{\exp(-2J_0)+1} \\ -1, & \text{with prob } \ge \frac{1}{\exp(-2J_0)+1} \end{cases} \tag{4.44}$$

Denote

$$Y_j \coloneqq \frac{X_j^{(t+1)} + 1}{2} \tag{4.45}$$

Note that $\{Y_j \mid j \in [N]\}$ are independent Bernoulli random variables.

By Lemma 4.3.4, $\forall r > 0$, with probability at least $1 - 2e^{-\frac{2r^2}{|C_G|}}$,

$$\begin{aligned}
\frac{1}{|C_G|} \sum_{j \in C_G} Y_j &< \mathbb{E}_{j \in C_G}[Y_j] + \frac{r}{|C_G|} \quad \text{(by Hoeffding's inequality Lemma 4.3.4)} \\
&\le \frac{\exp(-2J_0)}{\exp(-2J_0)+1} + \frac{r}{|C_G|} \quad \text{(by Equation (4.44) and definition of } Y_j \text{ in Equation (4.45))}
\end{aligned}$$

implying that with probability at least $1 - 2e^{-\frac{2r^2}{|C_G|}}$,

$$\frac{1}{|C_G|} \sum_{j \in C_G} X_j^{(t+1)} = 2 \frac{1}{|C_G|} \sum_{j \in C_G} Y_j - 1 < 2\left(\frac{\exp(-2J_0)}{\exp(-2J_0)+1} + \frac{r}{|C_G|}\right) - 1$$

i.e.

$$\sum_{j\in C_G} X_j^{(t+1)} < \frac{\exp\left(-2J_0\right)-1}{\exp\left(-2J_0\right)+1}\left|C_G\right| + 2r$$

Setting RHS to -2 solves to

$$r = -1 + \frac{1-\exp\left(-2J_0\right)}{\exp\left(-2J_0\right)+1}\frac{\left|C_G\right|}{2}$$

Hence

$$\text{with probability at least } 1-2e^{-\frac{2\left(-1+\frac{1-\exp\left(-2J_0\right)}{\exp\left(-2J_0\right)+1}\frac{\left|C_G\right|}{2}\right)^2}{\left|C_G\right|}}, \quad \sum_{j\in C_G} X_j^{(t+1)} < -2 \tag{4.46}$$

By union bound, $\forall T \in \mathbb{N}_+$,

$$\text{with probability at least } 1-2Te^{-\frac{2\left(-1+\frac{1-\exp\left(-2J_0\right)}{\exp\left(-2J_0\right)+1}\frac{\left|C_G\right|}{2}\right)^2}{\left|C_G\right|}}, \quad \forall t\in[T], \quad \sum_{j\in C_G} X_j^{(t)} < -2 \tag{4.47}$$

Note that when $\sum_{j\in C_G} X_j^{(t)} < -2$, $\boldsymbol{X}^{(t)} \notin \mathcal{R}_1$.

Finally, aligning the probabilities: setting

$$2Te^{-\frac{2\left(-1+\frac{1-\exp\left(-2J_0\right)}{\exp\left(-2J_0\right)+1}\frac{\left|C_G\right|}{2}\right)^2}{\left|C_G\right|}} = \delta$$

solves to

$$T = \frac{\delta}{2}e^{\frac{2\left(-1+\frac{1-\exp\left(-2J_0\right)}{\exp\left(-2J_0\right)+1}\frac{\left|C_G\right|}{2}\right)^2}{\left|C_G\right|}}$$

□

4.3.8 Proof of Corollary 4.1.1: Separation between N-Gibbs sampler and independent parallel sampling

This section provides additional information for the discussion at the end of Section 4.1.4.2.

Corollary 4.1.1 (Separation between N-Gibbs sampler and independent parallel sampling)**.** *On Ising model G in Equation (4.6) under Assumption 4.1.5, $\forall \delta \in (0,1)$, $\forall M \in \mathbb{N}_+$, If G additionally satisfies Assumption 4.1.6 and Assumption 4.1.7 and the initial $\boldsymbol{X}^{(0)}$ is such that $\sum_{i \in C_G} \boldsymbol{X}_i^{(0)} \leq -2$, then with probability at least $1-\delta$,*

1. *Running the k-**Gibbs sampler**: $\boldsymbol{X}^{(1)}_{k\text{-Gibbs}} \in \mathcal{R}_1$, and*
2. *Running **independent parallel**: $\{\boldsymbol{X}^{(t)}_{indep} | t \in [M]\} \cap \mathcal{R}_1 = \emptyset$*

Proof. Under the given conditions, with k**-Gibbs sampler**, by Proposition 4.1.4,

$$\text{with probability at least } 1 - \frac{\delta}{2}, \quad \{\boldsymbol{X}^{(t)}_{\text{k-Gibbs}} | t \in [\left\lceil \log_{c_{\mathcal{R}_1}} \frac{\delta}{2} \right\rceil]\} \cap \mathcal{R}_1 \neq \emptyset \tag{4.48}$$

in which the constant

$$c_{\mathcal{R}_1} \coloneqq 1 - \frac{\binom{N-|C_G|}{k-|C_G|}}{\binom{N}{k}} \frac{e^{2(J_0+h_G)}}{e^{2(J_0+h_G)} + e^{2J_0} + 2^{|C_G|} - 2} \tag{4.49}$$

Applying Assumption 4.1.6 to bound parts of the RHS of Equation (4.49):

$$\frac{e^{2J_0}}{e^{2(J_0+h_G)}} = e^{-2h_G}$$

$$\frac{2^{|C_G|}-2}{e^{2(J_0+h_G)}} \leq \frac{2^{|C_G|}}{e^{2(J_0+h_G)}} \leq \frac{2^{|C_G|}}{e^{|C_G|\ln 2 + 2h_G}} = \frac{2^{|C_G|}}{2^{|C_G|}e^{2h_G}} = e^{-2h_G}$$

Taking the sum:

$$\frac{e^{2J_0} + 2^{|C_G|} - 2}{e^{2(J_0+h_G)}} \leq 2e^{-2h_G}$$

Adding 1 to both sides:

$$\frac{e^{2(J_0+h_G)} + e^{2J_0} + 2^{|C_G|} - 2}{e^{2(J_0+h_G)}} \leq 1 + 2e^{-2h_G}$$

Taking the inverse:

$$\frac{e^{2(J_0+h_G)}}{e^{2(J_0+h_G)} + e^{2J_0} + 2^{|C_G|} - 2} \geq \frac{1}{1+2e^{-2h_G}} \geq \frac{1}{1+2e^{-2\frac{1}{2}\ln\frac{2(4-\delta)}{\delta}}} = \frac{1}{1+2\frac{\delta}{2(4-\delta)}} = \frac{1}{1+\frac{\delta}{4-\delta}} = \frac{4-\delta}{4} \tag{4.50}$$

Similarly, applying Assumption 4.1.7 to bound the other parts of the RHS of Equation (4.49):

$$\begin{aligned}
\frac{\binom{N-|C_G|}{k-|C_G|}}{\binom{N}{k}} &= \frac{\frac{(N-|C_G|)!}{(k-|C_G|)!(N-k)!}}{\frac{N!}{k!(N-k)!}} = \frac{\frac{(N-|C_G|)!}{(k-|C_G|)!}}{\frac{N!}{k!}} = \frac{(k+1-|C_G|)\cdots(N-|C_G|)}{(k+1)\cdots N} \\
&\geq \left(\frac{k+1-|C_G|}{k+1}\right)^{(N-k)} = \left(1 - \frac{|C_G|}{k+1}\right)^{(N-k)} \\
&\geq 1 - (N-k)\frac{|C_G|}{k+1} \quad (\text{since } \frac{|C_G|}{k+1} \in (0,1)\ ) \\
&\geq \frac{4-2\delta}{4-\delta} \quad (\text{by Assumption 4.1.7 and straightforward calculation})
\end{aligned}$$

Plugging in the above and Equation (4.50) to Equation (4.49):

$$c_{\mathcal{R}_1} \leq 1 - \frac{4-2\delta}{4-\delta} \cdot \frac{4-\delta}{4} = \frac{\delta}{2}$$

Plugging into Equation (4.48):

$$\text{with probability at least } 1 - \frac{\delta}{2}, \quad \{\boldsymbol{X}_{\text{k-Gibbs}}^{(t)} | t \in [1]\} \cap \mathcal{R}_1 \neq \emptyset, \quad \text{namely } \boldsymbol{X}_{\text{k-Gibbs}}^{(1)} \in \mathcal{R}_1 \tag{4.51}$$

On the other hand, with **independent parallel**, by Proposition 4.1.5,

$$\text{with probability at least } 1 - \frac{\delta}{2}, \quad \{\boldsymbol{X}_{indep}^{(t)} | t \in [\left\lfloor \frac{\delta}{4} \exp\left(c_{\text{stuck}}\right) \right\rfloor]\} \cap \mathcal{R}_1 = \emptyset \tag{4.52}$$

in which the constant

$$c_{\text{stuck}} \coloneqq \frac{2\left(-1 + \frac{1-\exp\left(-2J_0\right)}{\exp\left(-2J_0\right)+1} \frac{|C_G|}{2}\right)^2}{|C_G|} \tag{4.53}$$

Applying Assumption 4.1.6 to bound parts of the RHS:

$$\frac{1-\exp\left(-2J_0\right)}{\exp\left(-2J_0\right)+1} \geq \frac{1}{2}$$

Plugging into Equation (4.53):

$$\begin{aligned} c_{\text{stuck}} &\geq \frac{2\left(-1 + \frac{1}{2}\frac{|C_G|}{2}\right)^2}{|C_G|} \\ &= \frac{2\left(1 - \frac{|C_G|}{2} + \frac{|C_G|^2}{4}\right)}{|C_G|} \\ &\geq -1 + \frac{|C_G|}{8} \\ &\geq -1 + \left(1 + \ln \frac{4M}{\delta}\right) \quad \text{(by Assumption 4.1.6)} \\ &= \ln \frac{4M}{\delta} \end{aligned}$$

Plugging into Equation (4.52):

$$\text{with prob } \geq 1 - \frac{\delta}{2}, \quad \{\boldsymbol{X}_{indep}^{(t)} | t \in [\left\lfloor \frac{\delta}{4} \cdot \frac{4M}{\delta} \right\rfloor] = [M]\} \cap \mathcal{R}_1 = \emptyset, \quad \text{namely } \boldsymbol{X}_{indep}^{(t)} \notin \mathcal{R}_1 \forall t \in [M] \tag{4.54}$$

By union bound, with probability at least $1-\delta$, both Equation (4.51) and Equation (4.54) hold.

□

4.3.9 Background and proofs of Proposition 4.1.2 and Proposition 4.1.3: on the expressive power of Transformers for implementing sequence-to-sequence Markov chains in parallel

4.3.9.1 Technical setup and proofs

Background: Transformer network architecture. The transformer architecture (Vaswani et al., 2017) is a critical building block of many leading approaches to language modeling (Devlin et al., 2019; Brown et al., 2020). We refer the readers to these works for more details on the empirical promise that Transformer-based models have demonstrated. For theoretical understanding of Transformers, we refer the readers to prior works on their representational power (Yun et al., 2020; Yao et al., 2021; Liu et al., 2023a; Zhao et al., 2023), statistical sample complexity (Wei et al., 2021; Edelman et al., 2022), optimization process (Lu et al., 2021; Jelassi et al., 2022; Li et al., 2023), and interpretability (Wen et al., 2023), and references cited therein.

Mathematical setup. In the following we adapt and use the mathematical notations for the Transformer network architecture in Yun et al. (2020) and Li et al. (2023).

For each position of an input sequence (N tokens) $\tau_1 \cdots \tau_N$, use a d-dimensional *positional embedding* to represent that *position*, and use a d-dimensional *token embedding* for the *content* at that position. Hence, for the input sequence, both the token embeddings $\boldsymbol{E}$ and the positional embeddings $\boldsymbol{P}$ are matrices in $\mathbb{R}^{d\times N}$. Following empirical convention, the encoder function $g_{\text{enc}} : \Omega^N \mapsto \mathbb{R}^{d\times N}$ is

$$\boldsymbol{X} \coloneqq g_{\text{enc}}(\tau_1 \cdots \tau_N) = \boldsymbol{E} + \boldsymbol{P} \tag{4.55}$$

which form the input to the Transformer blocks:

A *Transformer block* $t^{h,m,r}$ (with h heads, head size m, and feed-forward hidden layer size r) is defined as

$$t^{h,m,r}(\boldsymbol{X}) \coloneqq \text{Attn}(\boldsymbol{X}) + \boldsymbol{W}_2 \cdot \text{ReLU}(\boldsymbol{W}_1 \cdot \text{Attn}(\boldsymbol{X}) + \boldsymbol{b}_1 \boldsymbol{1}_n^T) + \boldsymbol{b}_2 \boldsymbol{1}_n^T \tag{4.56}$$

where

$$\text{Attn}(\boldsymbol{X}) \coloneqq \boldsymbol{X} + \sum\nolimits_{i=1}^{h} \boldsymbol{W}_O^i \boldsymbol{W}_V^i \boldsymbol{X} \cdot \sigma[(\boldsymbol{W}_K^i \boldsymbol{X})^T \boldsymbol{W}_Q^i \boldsymbol{X}] \tag{4.57}$$

where the weight parameters $\boldsymbol{W}_O^i \in \mathbb{R}^{d\times m}$, $\boldsymbol{W}_V^i, \boldsymbol{W}_K^i, \boldsymbol{W}_Q^i \in \mathbb{R}^{m\times d}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d\times r}, \boldsymbol{W}_1 \in \mathbb{R}^{r\times d}, \boldsymbol{b}_2 \in \mathbb{R}^d, \boldsymbol{b}_1 \in \mathbb{R}^r$, and

$$\sigma : \mathbb{R}^{N_1\times N_2} \mapsto (0,1)^{N_1\times N_2}$$

is the column-wise softmax operation, such that

$$\sigma(A)_{ij} = \frac{\exp{(A_{ij})}}{\sum_{l=1}^{N} \exp{(A_{lj})}} \tag{4.58}$$

An L-layer Transformer is a composition of Transformer blocks:

$$\mathcal{T} \coloneqq \{g_{\text{transformer}} : \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N} \mid g_{\text{transformer}} = t_1 \circ \cdots \circ t_L \text{where } t_i \text{ is a Transformer block}\} \tag{4.59}$$

In the class of parallel decoding Transformers (denoted as $\mathcal{T}_{\text{PD}}$), for any Transformer $g_{\text{transformer}} \in \mathcal{T}$, its output $g_{\text{transformer}}(\boldsymbol{X}) \in \mathbb{R}^{d\times N}$ goes through a final affine transform and softmax (Equation (4.58)) to predict a distribution over tokens, for all positions

$$g_{\text{pred}}(\boldsymbol{X}) \coloneqq \sigma\left(\boldsymbol{W}^{\text{pred}} g_{\text{transformer}}(\boldsymbol{X}) + \boldsymbol{b}^{\text{pred}}\right) \in (0,1)^{|\Omega|\times N} \tag{4.60}$$

where $\boldsymbol{W}^{\text{pred}} \in \mathbb{R}^{|\Omega|\times d}$ and $\boldsymbol{b}^{\text{pred}} \in \mathbb{R}^{|\Omega|}$ are the prediction head weights and biases. Ω is the vocabulary of tokens.

For each position j, the predicted token τ_j is sampled from the predicted distribution $g_{\text{pred}}(\boldsymbol{X})_{:,j}$ *independently* with other positions

$$\tau_j \sim \texttt{sample}(g_{\text{pred}}(\boldsymbol{X})_{:,j}) \quad j \in [N] \tag{4.61}$$

where `sample` can be the standard sampling algorithm for multinomial distributions, or truncating the low-probability tail (Holtzman et al., 2020), or more conservatively, argmax sampling.

In the following definition, the sampling step can be denoted as a postprocessing function $s : (0,1)^{|\Omega|} \mapsto [0,1]^{|\Omega|}$ applied to each column of $g_{\text{pred}}(\boldsymbol{X})$. For example, argmax sampling s_{argmax} can be written as [25]:

$$\forall p \in (0,1)^{|\Omega|}, \quad s_{\text{argmax}}(p)_i = \begin{cases} 1, & \text{if } i = \min \arg\max_l p_l \\ 0, & \text{otherwise} \end{cases} \tag{4.62}$$

Thus, combining Equation (4.55) and Equation (4.60), a parallel decoding Transformer (denoted as $\mathcal{T}_{\text{PD}}$) and a sampling function s together define a Markov chain over sequences in Ω^N:

$$\mathcal{T}_{\text{PD}} \coloneqq \{g \coloneqq s \circ_c g_{\text{pred}} \circ g_{\text{enc}} : \Omega^N \to [0,1]^{|\Omega| \times N}\}. \tag{4.63}$$

where, again, s is understood as applying to each column of its input matrix separately. Specifically, in this Markov chain, $\forall g \in \mathcal{T}_{\text{PD}}$ the transition probabilities are

$$\mathbb{P}\left\{\tau_1 \cdots \tau_N, \tau_1' \cdots \tau_N'\right\} = \prod_{j \in [N]} g(\tau_1 \cdots \tau_N)_{\tau_j', j} \tag{4.64}$$

where the last $g(\tau_1 \cdots \tau_N)_{\tau_j', j}$ denotes the τ_j'-th row, j-th column of the matrix $g(\tau_1 \cdots \tau_N)$.

Yun et al. (2020) proved the following result on the expressivity of the Transformer network architecture:

Lemma 4.3.7 (Universal approximation by Transformers, informal (Yun et al., 2020))**.** *Let $1 \le p < \infty$ and $\epsilon > 0$, then for any compact set $\mathcal{D} \subset \mathbb{R}^{d \times n}$, for any given function $f : \mathcal{D} \mapsto \mathbb{R}^{d \times n}$, there exists a Transformer network $g \in \mathcal{T}^{2,1,4}$ of $O(N \left(\frac{1}{\delta}\right)^{dN})$ layers such that*

$$\left(\int \|f(\boldsymbol{X}) - g(\boldsymbol{X})\|_p^p \, d\boldsymbol{X}\right)^{1/p} \le \epsilon$$

in which δ is the smallest real number such that $\forall \boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{d \times n}$, if $\|\boldsymbol{X} - \boldsymbol{Y}\|_\infty < \delta$, then $\|f(\boldsymbol{X}) - f(\boldsymbol{Y})\|_p < \epsilon$. Moreover, the bound on the size of the constructed Transformer is asymptotically tight.

Lemma 4.3.8 (Transformers can simulate parallel solution to automata, Theorem 1 in (Liu et al., 2023a))**.** *Transformers can simulate the length-T output of all semiautomata with states Q, input alphabet Σ, and transition function $\delta : Q \times \Sigma \mapsto Q$. Moreover, the size of the simulating Transformer has depth $O(\log T)$, embedding dimension $O(|Q|)$, attention width $O(|Q|)$, and MLP width $O(|Q|^2)$.*

Remark 4.3.2. *Lemma 4.3.8 gives a more compact construction than a direct implication of more general universal approximation results Lemma 4.3.7 for Transformers.*

A direct corollary is Proposition 4.1.2:

Proposition 4.3.4 (Proposition 4.1.2 formalized)**.** *For any function $f : \Omega^N \mapsto \Omega^N$, [26] for any $T \in \mathbb{N}_+$, there exists a parallel decoding Transformer $g \in \mathcal{T}_{PD}$ (Equation (4.63)) whose sampling function is $s = s_{argmax}$ (Equation (4.62)), such that*

$$\forall \tau_1 \cdots \tau_N \in \Omega^N, \forall t \in [T], g^{(t)}(\tau_1 \cdots \tau_N) = f^{(t)}(\tau_1 \cdots \tau_N)$$

where the $f^{(t)}$ denotes composing the function f for t times.

[25] In Equation (4.62), the "min" stipulates that when multiple coordinates of p are tied for being argmax, i.e. $|\arg\max_l p_l| > 1$, then s_{argmax} would choose the smallest index in s_{argmax}. Other reasonable tie-breaking behaviors also work.

[26] This is equivalent to Markov chains over sequences in Ω^N whose transition probabilities are delta distributions $\mathbb{P}\left\{\tau_1 \cdots \tau_N, \tau_1' \cdots \tau_N'\right\} \in \{0, 1\}$. The correspondence is: for any Markov chain state $\tau_1 \cdots \tau_N$, $f(\tau_1 \cdots \tau_N)$ specifies the unique deterministic next state.

Proof. When each transition of a Markov chain is deterministic, i.e. if the next state distribution from any state is always a delta function, then the Markov chain reduces to a deterministic finite state automata, with states Ω^N, length N.

Applying Lemma 4.3.8, we get Transformers can simulate length-T output of this automata with depth $O(\log T)$, embedding dimension $O(|\Omega|^N)$, attention width $O(|\Omega|^N)$, and MLP width $O(|\Omega|^{2N})$. □

Proposition 4.3.5 (Proposition 4.1.3 formalized)**.** *Let Prod denote the set of Markov chains over sequences in Ω^N whose transition probabilities are product distributions over the positions, conditioned on the current state, i.e.* $\mathbb{P}\{\tau_1 \cdots \tau_N, \tau'_1 \cdots \tau'_N\} = \prod_{i=1}^{N} \mathbb{P}\{\tau'_i \mid \tau_1 \cdots \tau_N\}$. *Then,* $\mathcal{T}_{PD} = Prod$.

Proof. The statement involves both a positive result and a negative result on the expressivity of parallel decoding Transformers.

Positive: if the transition probability distribution is a product distribution conditioned on the current state, then the task of representing a Markov chain can be reduced to universally approximating a continuous function which maps all sequences to the correct logits $\boldsymbol{W}^{\text{pred}}\mathcal{T}(\boldsymbol{X}) + \boldsymbol{b}^{\text{pred}}$ in Equation (4.60), such that after softmax (Equation (4.58)) these logits produce the correct marginal distribution at each position. This is achievable by the construction in Lemma 4.3.7.

Negative: if the transition probability distribution is *not* a product distribution conditioned on the current state, then note that the sampling operations (Equation (4.61)) at positions j_1 and j_2 are *independent*, so Transformers cannot implement such Markov chains. □

Remark 4.3.3. *Proposition 4.1.3 implies that parallel decoding Transformers cannot* exactly represent *non-product distributions. However, it does not prevent parallel decoding Transformers from* approximating *some of non-product distributions. This is because certain non-product distributions can be approximated by product distributions.*

4.3.9.2 Connection to prior works in GMLM

Among existing language generation approaches via iterative refinement, Wang & Cho (2019) uses 1-Gibbs sampler. The approaches in Ghazvininejad et al. (2019); Savinov et al. (2022) and our experiments do not closely fall into either of *independent parallel* (Equation (4.4)) or the k-Gibbs sampler (Equation (4.3)) in Section 4.1.4. See Remark 4.3.4 for technical details.

Moreover, these approaches train models to learn the parameterized conditional distributions, which empirically may not admit a consistent joint distribution (Young & You, 2022; Torroba Hennigen & Kim, 2023).

To formally reason about the iterative refinement process in GMLMs, in Section 4.1.4 we relax some of these limitations to focus on several underlying theoretical obstacles that these methods face.

Remark 4.3.4 (Technical details in theoretically formalizing GMLM architectures)**.** *By Proposition 4.1.3, the sampling process in Ghazvininejad et al. (2019); Savinov et al. (2022) and our experiments are different from N-Gibbs sampler. Moreover, the sampling process is also different from* ***independent parallel*** *(Gibbs sampler 4.1.11): note that* ***independent parallel*** *strictly freezes all $X_{-\{i\}}^{(t)}$ when sampling*

$$X_i^{(t+1)} \sim p(\cdot \mid X_{-\{i\}}^{(t)})$$

whereas in Savinov et al. (2022) and our experiments, the model is trained to update all positions in parallel, which implies a different groundtruth next-iteration token distribution compared with $p(\cdot \mid X_{-\{i\}}^{(t)})$. In other words, although the updates are conditionally independent given the current state, the update probabilities are not trained to model $p(\cdot \mid X_{-\{i\}}^{(t)})$.

Mechanistically, Savinov et al. (2022) and our models in principle can take certain inter-position dependency into consideration (which ***independent parallel*** *cannot): for example, in layer L, position i can attend to* [27] *other positions e.g. j in the layer-(L − 1) representations. This enables the layer-L computation*

[27] via Transformer attention Equation (4.57)

at position i to be conditioned upon the intermediate representations at position j, which are not independent *from the final prediction at position j.*

Ghazvininejad et al. (2019) can be understood as predicting the subset of masked indices K in each update. The extent to which each update incorporates dependency between masked positions depends on implementation details: for example, whether attention masks are added to prevent any masked position from receiving attention.

4.3.10 Regularity conditions for asymptotic behavior of M-estimators

In the limite of infinite samples, M-estimators (in particular, maximum likelihood and the estimators in Definitions 4.1.2 and 4.1.7) converge in distribution to a normal distribution, under mild regularity conditions:

Lemma 4.3.9 (Van der Vaart (2000), Theorem 5.23; statement adapted from Qin & Risteski (2023))**.** *Consider a loss $L : \Theta \mapsto \mathbb{R}$, such that $L(\theta) = \mathbb{E}_p[\ell_\theta(x)]$ for $l_\theta : \mathcal{X} \mapsto \mathbb{R}$. Let Θ^* be the set of global minima of L, that is*

$$\Theta^* = \{\theta^* : L(\theta^*) = \min_{\theta \in \Theta} L(\theta)\}$$

Suppose the following conditions are met:

- *(Gradient bounds on l_θ) The map $\theta \mapsto l_\theta(x)$ is measurable and differentiable at every $\theta^* \in \Theta^*$ for p-almost every x. Furthermore, there exists a function $B(x)$, s.t. $\mathbb{E}\left[B(x)^2\right] < \infty$ and for every θ_1, θ_2 near θ^*, we have:*

$$|l_{\theta_1}(x) - l_{\theta_2}(x)| < B(x)\|\theta_1 - \theta_2\|_2$$

- *(Twice-differentiability of L) $L(\theta)$ is twice-differentiable at every $\theta^* \in \Theta^*$ with Hessian $\nabla^2_\theta L(\theta^*)$, and furthermore $\nabla^2_\theta L(\theta^*) \succ 0$.*
- *(Uniform law of large numbers) The loss L satisfies a uniform law of large numbers, that is*

$$\sup_{\theta \in \Theta} \left|\hat{\mathbb{E}}[l_\theta(x)] - L(\theta)\right| \xrightarrow{p} 0$$

- *(Realizability) The data distribution p satisfies: $\exists \theta^* \in \Theta$ such that $p_{\theta^*} = p$.*

Then, for every $\theta^* \in \Theta^*$, and every sufficiently small neighborhood S of θ^*, there exists a sufficiently large n, such that there is a unique minimizer $\hat{\theta}_n$ of $\hat{\mathbb{E}}[l_\theta(x)]$ in S. Furthermore, $\hat{\theta}_n$ satisfies:

$$\sqrt{n}(\hat{\theta}_n - \theta^*) \xrightarrow{d} \mathcal{N}\left(0, (\nabla^2_\theta L(\theta^*))^{-1} Cov(\nabla_\theta \ell(\theta^*; x))(\nabla^2_\theta L(\theta^*))^{-1}\right)$$

4.3.11 Convexity of pseudolikelihood for Ising models

Here, we expand on a comment in Section 4.1. We show that for a classic parametric class of distributions (namely, Ising models — which appear also in Section 4.1.4) the k-MPLE loss is in fact *convex*. This is a known fact which has been used to design (provably) efficient algorithms for learning bounded-degree Ising models (Ravikumar et al., 2010; Vuffray et al., 2016), and is just included for completeness. Recall the definition of Ising models from Equation (4.5) in Definition 4.1.12. Let Z be the partition function.

Proposition 4.3.6 (Fitting an Ising model over the conditional distributions is convex)**.** *When p_θ is an Ising model (Equation (4.5)), i.e. $\theta = (\boldsymbol{J}, \boldsymbol{h})$, the weighted pseudolikelihood objective (Definition 4.1.2) is convex.*

Proof. When p_θ is an Ising model (Equation (4.5)), we have:

$$\begin{aligned}
-\ln p_\theta(\mathbf{x}_K|\mathbf{x}_{-K}) &= -\ln \frac{\exp\left(\sum_{i\in[N]} \boldsymbol{h}_i \mathbf{x}_i + \sum_{i\neq j\in[N]} \boldsymbol{J}_{ij}\mathbf{x}_i\mathbf{x}_j\right)}{Z(\mathbf{x}_{-K})} \\
&= -\left(\sum_{i\in[N]} \boldsymbol{h}_i \mathbf{x}_i + \sum_{i\neq j\in[N]} \boldsymbol{J}_{ij}\mathbf{x}_i\mathbf{x}_j\right) + \ln Z(\mathbf{x}_{-K})
\end{aligned}$$

in which the denominator

$$\begin{aligned}
Z(\mathbf{x}_{-K}) = \sum_{X_K\in\{-1,1\}^{|K|}} \exp\Bigg(&\sum_{i\in K} \boldsymbol{h}_i X_K + \sum_{i\in[N]\setminus K} \boldsymbol{h}_i \mathbf{x}_i \\
&+ \sum_{i\neq j\in[K]} \boldsymbol{J}_{ij} X_i X_j + \sum_{i\in K, j\in[N]\setminus K} \boldsymbol{J}_{ij} X_i \mathbf{x}_j + \sum_{i\neq j\in[N]\setminus K} \boldsymbol{J}_{ij}\mathbf{x}_i\mathbf{x}_j\Bigg)
\end{aligned}$$

Note that $-\left(\sum_{i\in[N]} \boldsymbol{h}_i \mathbf{x}_i + \sum_{i\neq j\in[N]} \boldsymbol{J}_{ij}\mathbf{x}_i\mathbf{x}_j\right)$ is linear in $(\mathbf{h}, \mathbf{J})$ and $\ln Z(\mathbf{x}_{-K})$ is convex in (h, J), so $-\ln p_\theta(X_K = \mathbf{x}_K | X_{-K} = \mathbf{x}_{-K})$ is convex in $(\mathbf{h}, \mathbf{J})$, which completes the last piece of the proof.

□

4.4 Related works

Our theory is inspired by recent progress in sampling: the connections between pseudolikelihood and approximate tensorization of entropy are discussed in Marton (2013; 2015); Caputo et al. (2015); Caputo & Parisi (2021); Koehler et al. (2023). Benefits of k-Gibbs sampler are discussed in Lee (2023). Our experiments follow the framework that trains generative masked language models and generates samples using parallel decoding by iterative refinement: (Lee et al., 2018; Ghazvininejad et al., 2019; 2020; Kasai et al., 2020; Savinov et al., 2022), which tend to be at least twice faster than autoregressive approaches with a small drop in quality for tasks like machine translation. The inference process, which converts complete noise to full samples, might resemble diffusion models (Hoogeboom et al., 2021; Austin et al., 2021; Li et al., 2022; Gong et al., 2023; Zheng et al., 2023; Lou et al., 2024), but a key conceptual difference is that diffusion models are trained to revert a small amount of noise at each step, whereas the family of models that we study in this work are more similar to *masked autoencoders*: the training objective encourages reconstructing the whole target sequence in each step of decoding.

Non-autoregressive text generation Previous works applied various generative models to text, such as VAEs (Bowman et al., 2016; Bosc & Vincent, 2020), GANs (Che et al., 2017; Yu et al., 2017; Lin et al., 2017; Guo et al., 2018), and normalizing flows (Ziegler & Rush, 2019; Ma et al., 2019; Hoogeboom et al., 2021), but without a strong autoregressive component, the quality of generated text is often suboptimal. Later works achieve high-quality text generation through diffusion models (Hoogeboom et al., 2021; Austin et al., 2021; Li et al., 2022; Gong et al., 2023; Zheng et al., 2023) and energy-based models (Deng et al., 2020; Goyal et al., 2022; Qin et al., 2022), but their generation speeds tend to be much slower than autoregressive language models. Inference latency can be mitigated by approaches like Lee et al. (2020). Unlike the above paradigms that adapt continuous-domain generative models to text, our approach is closer to the following line of works that iteratively refine the generation process through parallel updates in the space of discrete token sequences, which tend to be at least twice faster than autoregressive approaches with a small drop in quality (Lee et al., 2018; Ghazvininejad et al., 2019; Stern et al., 2019; Guo et al., 2020; Ghazvininejad et al., 2020; Kasai et al., 2020; Savinov et al., 2022) (though autoregressive models also have the potential for speedup by using a shallower decoder for certain tasks (Kasai et al., 2021)). The generation quality of non-autoregressive models can be further improved by incorporating some autoregressive components (Kong et al., 2020; Reid et al., 2022) or input-output alignment (Chan et al., 2020; Saharia et al., 2020), or adaptive training curriculum (Qian et al., 2021). Insights such as the multimodality problem and components such as sequence-level knowledge distillation and input token fertility prediction were also proposed in (Gu et al., 2018). The benefit of distillation was verified in Kim & Rush (2016); Gu et al. (2018); Zhou et al. (2020); Gu & Kong (2021). Positional attention was tested in Gu et al. (2018); Kreutzer et al. (2020). Relevant to our experiments in Section 4.2.4, Ren et al. (2020) measure the target-side dependency as the proportion of attention paid to target tokens as opposed to the source tokens, in some modified attention architecture. Related to generation from MLMs, Wang & Cho (2019) use the learned conditionals inside a Gibbs sampler, but when the conditionals are not *consistent*, i.e. there is not a joint distribution that satisfies these conditionals, Gibbs sampler may amplify errors. In general, mathematical understanding about sampling from masked language models is still lagging substantially behind. Additionally, related to MLMs, Meng et al. (2023) analyzes some representational limitations, and Liu et al. (2022a) analyzes subtleties from a parameter identifiability view. Related to parallel decoding, recent work (Cai et al., 2024) parallelizes the inference with multiple heads by finetuning autoregressive LLM backbones.

Theory about parallel sampling Koehler et al. (2023) proved a generalization bound for pseudolikelihood estimator via the classic ($k = 1$) approximate tensorization of entropy, in the “proper learning” setting. Our generalization bound (Theorem 4.1.4) uses the generalized notion of the approximate tensorization of entropy (Definition 4.1.9), also apply to “improper learning” settings, and the proof involves quite different techniques. The classic approximate tensorization of entropy are discussed in Marton (2013; 2015); Caputo et al. (2015), which was more recently generalized to the “α-weighted block” version (Definition 4.1.9) in Caputo & Parisi (2021). Lee (2023) proves that k-Gibbs sampler mixes at least k times faster than 1-Gibbs sampler. For future works, recent algorithmic advances in parallel sampling could potentially be incorporated into our framework to achieve finer-grained theoretical analysis or better empirical quality-efficiency trade-off (Anari et al., 2023).

4.5 Conclusion

We introduce a new theoretical framework for understanding the power and limitations of generative masked language models (GMLM). In particular, our theory offers some guidance on the design spaces of learning and inference algorithms, through the perspectives of asymptotic sample complexity for parameter learning, finite-sample generalization bound for distribution learning, and the efficiency of Gibbs-like sampling algorithms. Empirically we adapt T5 to parallel decoding by iterative refinement (an non-autoregressive GMLM-based language generation strategy which showed strong speed-quality trade-off in the literature for tasks like machine translation). We recommend some rules of thumb for key design choices, and discuss the connection between the the empirical findings and our theory. For future works, we hope the theoretical framework and empirical observations can inspire new training objectives, inference algorithms, and neural network architectures better-suited for parallel decoding.

Chapter 5

Conclusion and future directions

In this thesis, I survey my research towards developing a methodology for theoretically reasoning about the data-training-inference interaction in language modeling: how specific structural properties of the data (such as semantic topics or syntactic recursion) are captured by the model during training, and how those learned features subsequently constrain or enable efficient inference algorithms.

By isolating key data structures—for example topic models (Blei et al., 2003) and context-free grammars (Dyck) (Schützenberger, 1963)—as theoretical sandboxes, we have moved beyond generalization bounds based on generic data distributions to provide mechanistic insights into how Transformers capture some simple linguistic structures. We then show that explicitly modeling certain structural properties of the data distribution also motivates useful settings for studying verifier-assisted language generation algorithms, and reveals fundamental connection between training and inference efficiencies in parallel non-autoregressive language models.

5.1 Summary of contributions

5.1.1 The mechanics of feature learning in Transformers (Chapter 2)

We investigated how Transformers learn simple linguistic structures. Moreover, we found that the flavor of the results depends sensitively on the type of structure we study.

In Section 2.1, we proved that for semantic structures modeled by topic models (Blei et al., 2003), the training dynamics are benign: simple one-layer Transformers naturally converge to interpretable solutions where token embeddings and attention patterns encode topic coherence.

However, Section 2.2 revealed a sharp contrast when the data structure shifts to context-free grammars (Dyck) (Schützenberger, 1963). We proved that in this case, the solution space is sufficient rich, and moreover, contrary to popular intuition, Transformers can achieve near-optimal generalization *without* learning interpretable, stack-like attention patterns. This effectively challenged "myopic" interpretability methods, showing that indistinguishable functional performance can arise from qualitatively different (and often uninterpretable) internal mechanisms.

5.1.2 Verifier-guided inference algorithms with backtracking (Chapter 3)

Since training does not always yield models that perfectly capture the structural constraints (as seen in the Dyck experiments in Section 2.2), we turned to study what inference algorithms better leverage the learned structures to generate samples that satisfy these structural constraints. We established a theoretical framework for *query complexity* in constrained generation, proving that access to a process verifier can render intractable generation tasks tractable (Section 3.1). Empirically, we demonstrated that a simple heuristic—Tokenwise Rejection Sampling with Backtracking (which allows the model to "erase" and regenerate tokens upon detecting a violation)—achieves favorable trade-off between accuracy, efficiency, and diversity (Section 3.2). Moreover, we propose a theoretically principled algorithm, value-guided sampling with stochastic backtracking, which provably mitigates error amplification as sequence length grows (Section 3.3).

5.1.3 Co-designing inference and training algorithms (Chapter 4)

Finally, we analyze Generative Masked Language Models (GMLMs), a parallel-efficient, non-autoregressive approach to language modeling. We established a fundamental link between the statistical efficiency of training and the computational efficiency of inference: both are governed by a quantity that measures the strength of cross-positional dependencies in the data distribution. We also applied our theoretical framework to prove that Transformers (commonly adopted in parallel decoding) cannot efficiently sample from distributions with strong cross-positional dependencies.

5.2 Outlook on future research

For many tasks that involve data-driven learning and inference, algorithmic advancements are typically demonstrated through evaluation metrics on various benchmarks. These benchmarks typically consist of many representative use cases of an algorithm, and estimate several aspects of the performance of the algorithm on these tasks. As researchers and practitioners develop more algorithms and test them on increasingly diverse benchmarks, they collectively expand the community's knowledge of these evaluation metrics at a rapid speed, and become more experienced in applying their knowledge to predict what modifications of their currently-investigated algorithm would improve these evaluation metrics. However, in many cases, it remains challenging to distill these vast and rapidly growing *knowledge* and *experience* into *systematic understanding* which satisfactorily compresses past observations and accurately informs future algorithmic design.

In this thesis, supported by my research, I argue that a promising methodology towards developing such systematic understanding is to focus on ***how the training and inference algorithms interact with a few key structures underlying the task data***.

First, identify and model relevant key structures of interest in the task data. Useful ways to identify these structures include investigating the predictions and error modes of the current algorithm, and improving our understanding of the task by thinking through all the steps by which data is generated. During these investigation, look for signals which suggest the current algorithm may systematically perform poorly in a certain type of cases, or overlooks a certain aspect of the task. Based on these signals, form more abstract hypotheses on what structures of the task data may be challenging for the current algorithm, and model a generative process of these structures.

Next, study how the training algorithm captures these structures, or how the inference algorithm leverages these structures. To make progress towards answering these questions, a conjectured generative process of these structures (which we modeled in the previous step and may refine later as we gather more evidence) may be helpful, because it allows us to reason about not just a few isolated occurrences of the structure of interest, but a distribution of cases to which this structure is relevant. In some cases, it may be illustrative to generate synthetic data based on our conjectured generative process of these structures, and run experiments which highlight how the algorithm interacts with this particular structure (while possibly ablating away some other structures in real data).

Finally, form and test concrete hypotheses about the data-training-inference interaction. These hypotheses may be about whether one type of algorithms is generally better than another type of algorithms in interacting with (e.g. capturing or leveraging) certain structures of the data, or proposing algorithmic improvements designed for better interacting with certain structures.

The goal of this line of research is to break down the task performance (measured by a large number of evaluation metrics on benchmarks) into capabilities defined by capturing or leveraging (a relatively small number of) key structures underlying the task data. Each key structure identified in this process provides a useful abstraction for studying the interaction between data and algorithms, enhances the researchers' and practitioners' understanding of the task data, and motivates the synergistic design of training and inference algorithms that better learn and leverage this structure.

Bibliography

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Zeyuan Allen-Zhu and Yuanzhi Li. Backward feature correction: How deep learning performs deep learning. *arXiv e-prints*, pp. arXiv–2001, 2020.

Nima Anari, Yizhi Huang, Tianyu Liu, Thuy-Duong Vuong, Brian Xu, and Katherine Yu. Parallel discrete sampling via continuous walks. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pp. 103–116, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399135. doi: 10.1145/3564246.3585207. URL https://doi.org/10.1145/3564246.3585207.

Sanjeev Arora, Rong Ge, Frederic Koehler, Tengyu Ma, and Ankur Moitra. Provable algorithms for inference in topic models. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2859–2867, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/arorab16.html.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=h7-XixPCAL.

Pranjal Awasthi and Andrej Risteski. On some provably correct cases of variational inference for topic models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/68a83eeb494a308fe5295da69428a507-Paper.pdf.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction, 2024.

Ainesh Bakshi, Allen Liu, Ankur Moitra, and Ewin Tang. High-temperature gibbs states are unentangled and efficiently preparable. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1027–1036. IEEE, 2024.

Maria-Florina Balcan, Avrim Blum, Zhiyuan Li, and Dravyansh Sharma. On learning verifiers for chain-of-thought reasoning. *arXiv preprint arXiv:2505.22650*, 2025.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, March 2022. doi: 10.1162/coli_a_00422. URL https://aclanthology.org/2022.cl-1.7.

Mikhail Belkin. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation, 2021. URL https://arxiv.org/abs/2105.14368.

Julian Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society Series D: The Statistician*, 24(3):179–195, 1975.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7096–7116, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.576. URL https://aclanthology.org/2020.emnlp-main.576.

Satwik Bhattamishra, Arkil Patel, and Navin Goyal. On the computational power of transformers and its implications in sequence modeling. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pp. 455–475, Online, November 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.conll-1.37. URL https://aclanthology.org/2020.conll-1.37.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=3X2EbBLNsk.

Blair Bilodeau, Natasha Jaques, Pang Wei Koh, and Been Kim. Impossibility theorems for feature attribution. *arXiv preprint arXiv:2212.11870*, 2022.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3 (null):993–1022, mar 2003. ISSN 1532-4435.

Adam Block, Dylan J Foster, Akshay Krishnamurthy, Max Simchowitz, and Cyril Zhang. Butterfly effects of SGD noise: Error amplification in behavior cloning and autoregression. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=CgPs04l9TO.

Sergey G Bobkov and Prasad Tetali. Modified logarithmic sobolev inequalities in discrete settings. *Journal of Theoretical Probability*, 19(2):289–336, 2006.

Ond rej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pp. 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2301.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ale s Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3302.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 conference on machine translation (WMT17). In Ondřej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, and Julia Kreutzer (eds.), *Proceedings of the Second Conference on Machine Translation*, pp. 169–214, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4717. URL https://aclanthology.org/W17-4717.

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv: Arxiv-2104.07143*, 2021.

Tom Bosc and Pascal Vincent. Do sequence-to-sequence VAEs learn global features of sentences? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4296–4318, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.350. URL https://aclanthology.org/2020.emnlp-main.350.

Edoardo Botta, Yuchen Li, Aashay Mehta, Jordan T. Ash, Cyril Zhang, and Andrej Risteski. On the query complexity of verifier-assisted language generation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9oIjvaDhoN.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 10–21, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1002. URL https://aclanthology.org/K16-1002.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4:1–43, 2012. URL https://api.semanticscholar.org/CorpusID:9316331.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. On identifiability in transformers. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJg1f6EFDB.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads, 2024.

Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, Ludwig Schubert, Chelsea Voss, Ben Egan, and Swee Kiat Lim. Thread: Circuits. *Distill*, 2020. doi: 10.23915/distill.00024. https://distill.pub/2020/circuits.

Pietro Caputo and Daniel Parisi. Block factorization of the relative entropy via spatial mixing. *Communications in Mathematical Physics*, 388(2):793–818, 2021.

Pietro Caputo, Georg Menz, and Prasad Tetali. Approximate tensorization of entropy at high temperature. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 24, pp. 691–716, 2015.

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. Imputer: Sequence modelling via imputation and dynamic programming. In *International Conference on Machine Learning*, pp. 1403–1413. PMLR, 2020.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks, 2017.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*, 2022a.

Valerie Chen, Jeffrey Li, Joon Sik Kim, Gregory Plumb, and Ameet Talwalkar. Interpretable machine learning: Moving from mythos to diagnostics. *Commun. ACM*, 65(8):43–50, jul 2022b. ISSN 0001-0782. doi: 10.1145/3546036. URL https://doi.org/10.1145/3546036.

Sehyun Choi, Tianqing Fang, Zhaowei Wang, and Yangqiu Song. Kcts: knowledge-constrained tree search decoding with token-level hallucination detection. *arXiv preprint arXiv:2310.09044*, 2023.

B. Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. *ARXIV.ORG*, 2023. doi: 10.48550/arXiv.2302.03025.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://aclanthology.org/W19-4828.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space, 2022.

Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. The efficiency misnomer. *arXiv preprint arXiv:2110.12894*, 2021.

Yichuan Deng, Zhihang Li, and Zhao Song. Attention scheme inspired softmax regression, 2023.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1l4SgHKDH.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Persi Diaconis and Laurent Saloff-Coste. Logarithmic sobolev inequalities for finite markov chains. *The Annals of Applied Probability*, 6(3):695–750, 1996.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Javid Ebrahimi, Dhruv Gelda, and Wei Zhang. How can self-attention networks recognize Dyck-n languages? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4301–4306, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.384. URL https://aclanthology.org/2020.findings-emnlp.384.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5793–5831. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/edelman22a.html.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english?, 2023.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM.

Dylan J Foster, Zakaria Mhammedi, and Dhruv Rohatgi. Is a good foundation necessary for efficient reinforcement learning? the computational role of the base model in exp˘loration. *arXiv preprint arXiv:2503.07453*, 2025.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference On Learning Representations*, 2018.

Yeqi Gao, Sridhar Mahadevan, and Zhao Song. An over-parameterized exponential regression, 2023.

F. Gers and J. Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE transactions on neural networks*, 12 6:1333–40, 2001.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1633. URL https://aclanthology.org/D19-1633.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. Semi-autoregressive training improves mask-predict decoding, 2020.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=jQj-_rLVXsj.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. Exposing the implicit energy networks behind masked language models via metropolis–hastings. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=6PvWo1kEvlT.

Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

Christopher Grimsley, Elijah Mayfield, and Julia R.S. Bursten. Why attention is not explanation: Surgical intervention and causal reasoning about neural models. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 1780–1790, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.220.

Jiatao Gu and Xiang Kong. Fully non-autoregressive neural machine translation: Tricks of the trade. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 120–133, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.11. URL https://aclanthology.org/2021.findings-acl.11.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Nature 645, 633–638*, 2025.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Junliang Guo, Linli Xu, and Enhong Chen. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 376–385, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.36. URL https://aclanthology.org/2020.acl-main.36.

Jonathan Haab, Nicolas Deutschmann, and María Rodríguez Martínez. Is attention interpretation? a quantitative assessment on sets. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pp. 303–321, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-23618-1.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Trans. Assoc. Comput. Linguistics*, 8:156–171, 2020. doi: 10.1162/tacl_a_00306. URL https://doi.org/10.1162/tacl_a_00306.

Jaroslav Hájek. Local asymptotic minimax and admissibility in estimation. In *Proceedings of the sixth Berkeley symposium on mathematical statistics and probability*, volume 1, pp. 175–194, 1972.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

P Hayes-Roth, M Fox, G Gill, DJ Mostow, and R Reddy. Speech understanding systems: Summary of results of the five-year research effort, 1976.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2733–2743, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1275. URL https://aclanthology.org/D19-1275.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4129–4138, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1419. URL https://www.aclweb.org/anthology/N19-1419.

John Hewitt, Michael Hahn, Surya Ganguli, Percy Liang, and Christopher D. Manning. RNNs can generate bounded hierarchical languages with optimal memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1978–2010, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.156. URL https://www.aclweb.org/anthology/2020.emnlp-main.156.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=6nbpPqUCIi7.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. Do attention heads in bert track syntactic dependencies? *ArXiv*, abs/1911.12246, 2019.

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. *International Conference on Learning Representations (ICLR)*, 2025a.

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. *International Conference on Machine Learning (ICML)*, 2025b.

Fuchun Huang and Yosihiko Ogata. Generalized pseudo-likelihood estimates for markov random fields on lattice. *Annals of the Institute of Statistical Mathematics*, 54:1–18, 2002.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3543–3556, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1357. URL https://aclanthology.org/N19-1357.

Samy Jelassi, Michael Eli Sander, and Yuanzhi Li. Vision transformers provably learn spatial structure. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=eMW9AkXaREI.

Mark Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43(2–3):169–188, 1986. doi: 10.1016/0304-3975(86)90174-X.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. Unsupervised grammar induction with depth-bounded pcfg. *Transactions of the Association for Computational Linguistics*, 6:211–224, 2018.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Daniel Jurafsky and James H Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, 2000.

Fred Karlsson. Constraints on multiple center-embedding of clauses. *Journal of Linguistics*, 43(2):365–392, 2007.

Richard Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, volume 40, pp. 85–103, 01 1972. ISBN 978-3-540-68274-5. doi: 10.1007/978-3-540-68279-0_8.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. Non-autoregressive machine translation with disentangled context transformer. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah Smith. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KpfasTaLUpq.

Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, 01 2004. ISBN 978-3-540-40286-2. doi: 10.1007/978-3-540-24777-7.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M. Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=DjJmre5IkP.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7057–7075, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.574. URL https://aclanthology.org/2020.emnlp-main.574.

Frederic Koehler, Alexander Heckett, and Andrej Risteski. Statistical efficiency of score matching: The view from isoperimetry. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TD7AnQjNzR6.

Xiang Kong, Zhisong Zhang, and Eduard Hovy. Incorporating a local translation mechanism into non-autoregressive translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1067–1073, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.79. URL https://aclanthology.org/2020.emnlp-main.79.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4365–4374, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1445. URL https://aclanthology.org/D19-1445.

Julia Kreutzer, George Foster, and Colin Cherry. Inference strategies for machine translation with conditional masking. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5774–5782, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.465. URL https://aclanthology.org/2020.emnlp-main.465.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Holden Lee. Parallelising glauber dynamics, 2023.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1173–1182, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1149. URL https://aclanthology.org/D18-1149.

Jason Lee, Raphael Shu, and Kyunghyun Cho. Iterative refinement in the continuous space for non-autoregressive neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1006–1015, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.73. URL https://aclanthology.org/2020.emnlp-main.73.

David A Levin, Yuval Peres, and Elizabeth L Wilmer. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2009.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*, 2024a.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 681–691, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1082. URL https://aclanthology.org/N16-1082.

Xian Li and Hongyu Gong. Robust optimization for multilingual translation with imbalanced data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 25086–25099. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/d324a0cc02881779dcda44a675fdcaaa-Paper.pdf.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-LM improves controllable text generation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=3s9IrEsjLyk.

Yuanzhi Li, Tengyu Ma, and Hongyang R Zhang. Learning over-parametrized two-layer neural networks beyond ntk. In *Conference on learning theory*, pp. 2613–2682. PMLR, 2020.

Yuchen Li and Andrej Risteski. The limitations of limited context for constituency parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2675–2687, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.208. URL https://aclanthology.org/2021.acl-long.208.

Yuchen Li, Yuanzhi Li, and Andrej Risteski. How do transformers learn topic structure: Towards a mechanistic understanding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Yuchen Li, Alexandre Kirchmeyer, Aashay Mehta, Yilong Qin, Boris Dadachev, Kishore Papineni, Sanjiv Kumar, and Andrej Risteski. Promises and pitfalls of generative masked language modeling: Theoretical framework and practical guidelines. In *Forty-first International Conference on Machine Learning*, ICML'24. JMLR.org, 2024b.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Chu-Cheng Lin, Aaron Jaech, Xin Li, Matthew R. Gormley, and Jason Eisner. Limitations of autoregressive models and their alternatives. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5147–5173, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.405. URL https://aclanthology.org/2021.naacl-main.405.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 3158–3168, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Yongjie Lin, Yi Chern Tan, and Robert Frank. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 241–253, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4825. URL https://aclanthology.org/W19-4825.

Zachary C. Lipton. The mythos of model interpretability, 2017.

Bingbin Liu, Daniel Hsu, Pradeep Kumar Ravikumar, and Andrej Risteski. Masked prediction: A parameter identifiability view. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=Hbvlb4D1aFC.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=De4FYqjFueZ.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Exposing attention glitches with flip-flop language modeling. *arXiv preprint arXiv:2306.00946*, 2023b.

Emmy Liu and Graham Neubig. Are representations built from the ground up? an empirical examination of local composition in language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9053–9073, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.617. URL https://aclanthology.org/2022.emnlp-main.617.

Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: implicit bias matters for language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023c.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. In *First Conference on Language Modeling*, 2024.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5747–5763, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.463. URL https://aclanthology.org/2020.emnlp-main.463.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1073–1094, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1112. URL https://aclanthology.org/N19-1112.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022b.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Bruce P Lowerre and B Raj Reddy. Harpy, a connected speech recognition system. *The Journal of the Acoustical Society of America*, 59(S1):S97–S97, 1976.

Haoye Lu, Yongyi Mao, and Amiya Nayak. On the dynamics of training attention models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=1OCTOShAmqB.

Zeping Luo, Cindy Weng, Shiyou Wu, Mo Zhou, and Rong Ge. One objective for all models–self-supervised learning for topic models. *arXiv preprint arXiv:2203.03539*, 2022.

Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*, 2023.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4282–4292, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1437. URL https://aclanthology.org/D19-1437.

Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pp. 6682–6691. PMLR, 2020.

Katalin Marton. An inequality for relative entropy and logarithmic sobolev inequalities in euclidean spaces. *Journal of Functional Analysis*, 264(1):34–61, 2013.

Katalin Marton. Logarithmic sobolev inequalities in discrete product spaces: a proof by a transportation cost distance. *arXiv preprint arXiv:1507.02803*, 2015.

Clara Meister, Stefan Lazov, Isabelle Augenstein, and Ryan Cotterell. Is sparse attention more interpretable? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 122–129, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.17. URL https://aclanthology.org/2021.acl-short.17.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Yu Zhang, and Jiawei Han. Topic discovery via latent space clustering of pretrained language model representations. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 3143–3152, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512034. URL https://doi.org/10.1145/3485447.3512034.

Yu Meng, Jitin Krishnan, Sinong Wang, Qifan Wang, Yuning Mao, Han Fang, Marjan Ghazvininejad, Jiawei Han, and Luke Zettlemoyer. Representation deficiency in masked language modeling. *arXiv preprint arXiv:2302.02060*, 2023.

William Merrill. Sequential neural networks as automata. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pp. 1–13, Florence, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3901. URL https://www.aclweb.org/anthology/W19-3901.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf.

Ankur Moitra. *Algorithmic aspects of machine learning*. Cambridge University Press, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022. URL https://arxiv.org/abs/2112.09332.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Toan Q. Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong, November 2-3 2019. Association for Computational Linguistics. URL https://aclanthology.org/2019.iwslt-1.17.

Andrew M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Proceedings of Symposia in Applied Mathematics*, 1998. URL https://api.semanticscholar.org/CorpusID:115995195.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads, 2022. URL https://arxiv.org/abs/2209.11895.

Peng Si Ow and Thomas E Morton. Filtered beam search in scheduling. *The International Journal Of Production Research*, 26(1):35–62, 1988.

Chirag Pabbaraju, Dhruv Rohatgi, Anish Sevekari, Holden Lee, Ankur Moitra, and Andrej Risteski. Provable benefits of score matching. *arXiv preprint arXiv:2306.01993*, 2023.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris Papailiopoulos. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. *Advances in neural information processing systems*, 33:2599–2610, 2020.

Jorge Perez, Pablo Barcelo, and Javier Marinkovic. Attention is turing-complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021. URL http://jmlr.org/papers/v22/20-302.html.

Thomas Plantard, Willy Susilo, and Zhenfei Zhang. Lattice reduction for modular knapsack. In *Selected Areas in Cryptography: 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers 19*, pp. 275–286. Springer, 2013.

Ashwini Pokle, Jinjin Tian, Yuchen Li, and Andrej Risteski. Contrasting the landscape of contrastive and non-contrastive learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 8592–8618. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/pokle22a.html.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. When BERT Plays the Lottery, All Tickets Are Winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3208–3229, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.259. URL https://aclanthology.org/2020.emnlp-main.259.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 157–163, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2025.

Amy Pu, Hyung Won Chung, Ankur P. Parikh, Sebastian Gehrmann, and Thibault Sellam. Learning compact metrics for MT. *CoRR*, abs/2110.06341, 2021. URL https://arxiv.org/abs/2110.06341.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. Glancing transformer for non-autoregressive neural machine translation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1993–2003, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.155. URL https://aclanthology.org/2021.acl-long.155.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. COLD decoding: Energy-based constrained text generation with langevin dynamics. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TiZYrQ-mPup.

Yilong Qin and Andrej Risteski. Fit like you sample: Sample-efficient generalized score matching from fast mixing diffusions, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Alessandro Raganato and Jörg Tiedemann. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 287–297, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5431. URL https://aclanthology.org/W18-5431.

Pradeep Ravikumar, Martin J Wainwright, and John D Lafferty. High-dimensional ising model selection using l1-regularized logistic regression. *Ann. Statist. 38(3): 1287-1319 (June 2010). DOI: 10.1214/09-AOS691*, 2010.

Machel Reid, Vincent Josua Hellendoorn, and Graham Neubig. Diffuser: Diffusion via edit-based reconstruction. In *The Eleventh International Conference on Learning Representations*, 2022.

Yi Ren, Jinglin Liu, Xu Tan, Zhou Zhao, Sheng Zhao, and Tie-Yan Liu. A study of non-autoregressive model for sequence generation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 149–159, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.15. URL https://aclanthology.org/2020.acl-main.15.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*, 2022. URL https://arxiv.org/abs/2203.17189.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2021.

Dhruv Rohatgi, Abhishek Shetty, Donya Saless, Yuchen Li, Ankur Moitra, Andrej Risteski, and Dylan J. Foster. Taming imperfect process verifiers: A sampling perspective on backtracking, 2025. URL https://arxiv.org/abs/2510.03149.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. Non-autoregressive machine translation with latent alignments. *arXiv preprint arXiv:2004.07437*, 2020.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=T0GpzBQ1Fg6.

Robin M. Schmidt, Telmo Pires, Stephan Peitz, and Jonas Lööf. Non-autoregressive neural machine translation: A call for clarity, 2022.

M.P. Schützenberger. On context-free languages and push-down automata. *Information and Control*, 6 (3):246–264, 1963. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(63)90306-1. URL https://www.sciencedirect.com/science/article/pii/S0019995863903061.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. BLEURT: learning robust metrics for text generation. *CoRR*, abs/2004.04696, 2020. URL https://arxiv.org/abs/2004.04696.

Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2931–2951, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1282. URL https://aclanthology.org/P19-1282.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235, 2018. URL http://arxiv.org/abs/1804.04235.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

Suzanna Sia, Ayush Dalmia, and Sabrina J. Mielke. Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too! In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1728–1736, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.135. URL https://aclanthology.org/2020.emnlp-main.135.

Hava T. Siegelmann and Eduardo D. Sontag. On the computational power of neural nets. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pp. 440–449, New York, NY, USA, 1992. Association for Computing Machinery. ISBN 089791497X. doi: 10.1145/130385.130432. URL https://doi.org/10.1145/130385.130432.

Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93–133, 1989.

Charlie Snell, Ruiqi Zhong, Dan Klein, and Jacob Steinhardt. Approximating how single head attention learns, 2021.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

David Sontag and Dan Roy. Complexity of inference in latent dirichlet allocation. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/3871bd64012152bfb53fdf04b401193f-Paper.pdf.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pp. 5976–5985. PMLR, 2019.

Kaiser Sun and Ana Marasović. Effective attention sheds light on interpretability. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 4126–4135, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.361. URL https://aclanthology.org/2021.findings-acl.361.

Xiaobing Sun and Wei Lu. Understanding attention for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3418–3428, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.312. URL https://aclanthology.org/2020.acl-main.312.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. LSTM networks can perform dynamic counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pp. 44–54, Florence, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3905. URL https://www.aclweb.org/anthology/W19-3905.

Mozhgan Talebpour, Alba García Seco de Herrera, and Shoaib Jameel. Topics in contextualised attention embeddings. In Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (eds.), *Advances in Information Retrieval*, pp. 221–238, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-28238-6.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1452. URL https://aclanthology.org/P19-1452.

Laure Thompson and David Mimno. Topic modeling with contextualized word representation clusters, 2020.

Alexis Akira Toda. Operator reverse monotonicity of the inverse. *The American Mathematical Monthly*, 118 (1):82–83, 2011.

Lucas Torroba Hennigen and Yoon Kim. Deriving language models from masked language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1149–1159, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.99. URL https://aclanthology.org/2023.acl-short.99.

Christopher Tosh, Akshay Krishnamurthy, and Daniel Hsu. Contrastive estimation reveals topic posterior information to linear models. *J. Mach. Learn. Res.*, 22(1), jan 2021. ISSN 1532-4435.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. An empirical study on robustness to spurious correlations using pre-trained language models. *Transactions of the Association for Computational Linguistics*, 8:621–633, 2020. doi: 10.1162/tacl_a_00335. URL https://aclanthology.org/2020.tacl-1.40.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Shubham Ugare, Tarun Suresh, Hangoo Kang, Sasa Misailovic, and Gagandeep Singh. Syncode: Llm generation with grammar augmentation. *arXiv preprint arXiv:2403.01632*, 2024.

Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 63–76, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4808. URL https://aclanthology.org/W19-4808.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580.

Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=rvZv7sDPV9.

Marc Vuffray, Sidhant Misra, Andrey Lokhov, and Michael Chertkov. Interaction screening: Efficient and sample-optimal learning of ising models. *Advances in neural information processing systems*, 29, 2016.

Alex Wang and Kyunghyun Cho. BERT has a mouth, and it must speak: BERT as a Markov random field language model. In Antoine Bosselut, Asli Celikyilmaz, Marjan Ghazvininejad, Srinivasan Iyer, Urvashi Khandelwal, Hannah Rashkin, and Thomas Wolf (eds.), *Proceedings of the Workshop on Methods for*

Optimizing and Evaluating Neural Language Generation, pp. 30–36, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-2304. URL https://aclanthology.org/W19-2304.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Colin Wei, Yining Chen, and Tengyu Ma. Statistically meaningful approximation: a case study on approximating turing machines with transformers, 2021. URL https://arxiv.org/abs/2107.13163.

Gail Weiss, Yoav Goldberg, and Eran Yahav. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 740–745, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2117. URL https://www.aclweb.org/anthology/P18-2117.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11080–11090. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/weiss21a.html.

Kaiyue Wen, Yuchen Li, Bingbin Liu, and Andrej Risteski. Transformers are uninterpretable with myopic methods: a case study with bounded dyck grammars. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=OitmaxSAUu.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 11–20, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1002. URL https://aclanthology.org/D19-1002.

WikimediaFoundation. Wikimedia downloads. *Wikimedia Downloads*, 2023. URL https://dumps.wikimedia.org.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4166–4176, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.383. URL https://aclanthology.org/2020.acl-main.383.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36, 2024.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Kevin Yang and Dan Klein. FUDGE: Controlled text generation with future discriminators. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3511–3535, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.276. URL https://aclanthology.org/2021.naacl-main.276/.

Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Ding-Chu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. Step back to leap forward: Self-backtracking for boosting reasoning of language models. *arXiv preprint arXiv:2502.04404*, 2025.

Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pp. 222–227. IEEE Computer Society, 1977.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3770–3785, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.292. URL https://aclanthology.org/2021.acl-long.292.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2023.

Tom Young and Yang You. On the inconsistencies of conditionals learned by masked language models. *arXiv preprint arXiv:2301.00068*, 2022.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pp. 2852–2858. AAAI Press, 2017.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ByxRM0Ntvr.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15383–15393. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b05b57f6add810d3b7490866d74c0053-Paper.pdf.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023a.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*, 2023b.

Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. Unveiling transformers with lego: a synthetic reasoning task, 2022a. URL https://arxiv.org/abs/2206.04301.

Yufeng Zhang, Boyi Liu, Qi Cai, Lingxiao Wang, and Zhaoran Wang. An analysis of attention via the lens of exchangeability and latent variable models, 2023c.

Zihan Zhang, Meng Fang, Ling Chen, and Mohammad Reza Namazi Rad. Is neural topic modelling better than clustering? an empirical study on clustering with contextual embeddings for topics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3886–3893, Seattle, United States, July 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.285. URL https://aclanthology.org/2022.naacl-main.285.

Haoyu Zhao, Abhishek Panigrahi, Rong Ge, and Sanjeev Arora. Do transformers parse while predicting the masked word?, 2023.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language models via twisted sequential monte carlo. *arXiv preprint arXiv:2404.17546*, 2024.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation, 2023.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S5wmbQc1We.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Chunting Zhou, Jiatao Gu, and Graham Neubig. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BygFVAEKDH.

Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In *International Conference on Machine Learning*, pp. 7673–7682. PMLR, 2019.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.