Ph.D. Thesis

Learning with Auxiliary Information: from Language to Tables

Juyong Kim

November 2025

CMU-ML-25-119

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

Thesis Committee:
Pradeep Ravikumar (Chair)
Jeremy Weiss (National Institute of Health)
Rayid Ghani
George H. Chen

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

This research was supported by: Air Force Research Laboratory award FA87502321015; Office of Naval Research award N000141812861; National Science Foundation awards DMS1661802 and OAC1934584; Unites States Army award W911NF1720196; and gifts from Optum and the University of Pittsburgh Medical Center.

Keywords: Auxiliary information, Natural Language Processing, Tabular Data Learning, Knowledge-Enriched Learning

Abstract

Over the past decade, deep-learning models have achieved remarkable progress across many machine learning tasks, largely by training ever larger architectures on massive, fairly homogenized datasets. In practice, however, these models often ignore the "extra" information that accompanies real-world data and rely exclusively on the primary data modality for generalization. This thesis argues that such auxiliary signals can be systematically leveraged to improve accuracy, data efficiency, and interpretability.

The first part of the thesis focuses on natural language processing and reports three studies. I show that (i) a simple symbolic-statistical model, combining a character-level language model with a clinical lexicon, can improve spelling correction in noisy clinical notes, (ii) structural attention masks derived from parse trees help Transformers handle hard compositional generalization splits, and (iii) linking discharge summaries to time-stamped electronic health record tables refines clinical event timelines.

The second part turns to tabular data, where gradient-boosted trees still dominate many benchmarks. First, I introduce a knowledge-enriched framework that injects deterministic information into tabular learners, define concept kernels that encode relations between columns, and release a public benchmark of tabular datasets equipped with column descriptions, embeddings, and concept kernels. Building on this, I propose a concept-conditioned tabular model that represents each cell as a function of both its value and a semantic embedding of the corresponding column, and pretrain it across an extended collection of datasets. Together, these contributions aim to clarify how auxiliary domain knowledge can be turned into effective inductive biases.

For the love of my life, Jisoo.

Acknowledgements

Being part of CMU's Ph.D. program has been one of the most meaningful experiences of my life. At the same time, pursuing a Ph.D. has been a long journey of discovering myself and developing the discipline and resilience needed to move forward. None of this would have been possible without the support, guidance, and encouragement of many remarkable people.

First and foremost, I would like to thank my advisors, Pradeep Ravikumar and Jeremy Weiss, for their invaluable mentorship throughout my time at CMU. Their insight, patience, and steady encouragement sustained me through every stage of this process. I am also deeply grateful to my committee members, Rayid Ghani and George Chen, for their thoughtful feedback and support.

Throughout my Ph.D., I had the privilege of interacting and collaborating with many inspiring researchers, including Linyuan Gong, Justin Khim, Joshua Ainslie, Santiago Ontañón, Abheesht Sharma, Suhas Shanbhogue, Wenbin Zhang, Cheng Cheng, Diego Salazar, Soyong Shin, Chandler Squires, Zihao Ye, Johnna Sundberg, Burak Varıcı, and Daniel Jeong. Each of them shaped the way I think about research, and I am sincerely thankful for the opportunity to learn from and work with them.

I am also grateful for the many colleagues and friends in the Machine Learning Department: Che-ping Tsai, Conor Igor, Ian Char, Jisu Kim, Joon Sik Kim, Kwangho Kim, Rattana Pukdee, Xun Zheng, Youngseog Chung, and many others, who, in different moments, brought joy, comfort, and motivation. Their presence made the demanding stretches of this journey feel lighter, and their curiosity and dedication continually inspired me.

Living far from home could easily have been an isolating experience, but it never felt that way thanks to the friends with whom I shared everyday life. Byeongjoo Ahn, Soyong Shin, Haejoon Lee, Jinmo Rhee, Byeongsoo Jeon, Kiwan Maeng, Hun Namkung, and the 93 group, along with many others, made unfamiliar places feel familiar and helped transform challenges into memories I will always cherish. If I have forgotten to mention anyone, I sincerely apologize. It is only due to my poor memory and not a lack of gratitude.

Finally, my deepest appreciation goes to my family and to my partner, Jisoo, for their unconditional love and support, which underlies everything I do. This work is dedicated to them.

Contents

List of Figures

List of Tables

1 Introduction

Recent advances in deep learning have been driven by large-scale models trained on massive, mostly homogeneous data. However, these models typically overlook the rich *auxiliary signals* that accompany real-world datasets—domain lexicons, syntactic parses, column descriptions, and other deterministic metadata. Leveraging such side information offers three potential advantages: higher predictive accuracy, improved data efficiency, and greater interpretability. This proposal investigates auxiliary signals in two complementary settings:

1) In natural-language processing, where large language models already achieve strong baselines, I examine whether targeted cues can yield measurable gains.

2) In tabular learning, a domain where gradient-boosted trees remain dominant, I study how explicit column knowledge can be formalized as concept kernels and can enable multi-table training.

By integrating these signals into both language and tabular pipelines, the work aims to provide practical methods for pushing neural models beyond purely data-driven training.

Overview of Thesis The first part of the thesis focuses on natural language processing and presents three studies. (1) For clinical spelling correction, I build a symbolic-statistical model that combines a character-level language model with a simple corruption rule and narrows the search space using a clinical vocabulary. (2) For compositional generalization, I add structural attention masks derived from parse trees, which helps Transformer to solve the hardest splits. (3) For clinical event timeline prediction, I design a multimodal model that links events in discharge notes with time-stamped tables from EHR. These results show that symbolic knowledge, structure, and cross-modal links are useful signal for NLP.

The second part of the thesis turns to tabular machine learning, where tree-based models such as XGBoost are still set the standard. I introduce a *knowledge-enriched* framework that injects deterministic side information into tabular models. As a first instantiation, I define *kernel-enriched* supervised learning, in which a *concept kernel*, a kernel over columns encoding their semantic relationships, supplements the usual training data. To make this line of work reproducible, I release KE-TALENT, a benchmark of 11 public datasets, each bundled with column descriptions, embedding-based concept kernels, and training pipelines. Building on this foundation, I then develop a *concept-conditioned tabular model* that represents each cell as a function of both its value and a semantic embedding of the corresponding column, and I explore whether such a model can be pre-trained across multiple tables using masked value prediction and contrastive objectives before being fine-tuned on individual datasets.

Across both parts, the aim is to understand when and how auxiliary information about columns can be translated into useful inductive biases for neural models, and to provide concrete recipes for combining data-driven learning with richer side information in both NLP and tabular settings.

Part I

Learning with Auxiliary Information in NLP

2 A Symbolic–Statistical Approach for Clinical Spelling Correction

This section is based on Kim et al. [2022].

2.1 Introduction

Spelling correction is an old problem in natural language processing and is especially essential in healthcare environments that are rife with error-prone text. Estimates of spelling error rates in clinical notes range from 0.4% [Lai et al., 2015] to 7% [Tolentino et al., 2007]. Correcting misspelled words in clinical texts is crucial since misspellings can have a notorious effect on downstream NLP tasks such as automatic diagnostic coding (assigning diagnosis codes given the clinical note).

Such spelling correction in a clinical context has several challenges. First, the candidate generation step, a critical component of most spelling correction methods, requires time complexity proportional to the vocabulary size. Second, the context representations, such as n-gram and word embeddings, of context-sensitive methods cannot properly handle rare words or words not in the embedding dictionary. In healthcare settings, both issues above are more severe than in general English text because of more extensive and specialized clinical vocabulary.

In this work, we propose a probabilistic model of correcting misspellings, named Conditional Independence Model (CIM), where we compute the posterior probability of the correct word given the misspelled word and the context. We assume that the misspelling is independent of the context given the correct word, and under this assumption, the posterior probability can be decomposed into a language model component and a corruption model component. Using a character-level language model and a simple edit-based corruption model, we can naturally decode the correct word using beam search.

This simple approach addresses both of the caveats raised earlier. CIM generates output candidates in a computationally inexpensive auto-regressive manner. And the deep language model provides the probability of any candidate given the context, which solves the issue of rare or out-of-embedding words. Moreover, our approach is modular, and allows for incorporating any advances in both language models, as well as edit-based corruption models. We validate the effectiveness of CIM with the two healthcare misspelling datasets and show CIM can be trained and tuned with fully unsupervised settings. To our knowledge, this work is the first approach to the spelling correction problem with the noisy channel model combined with a deep character-level language model.

2.2 Related Works

Spelling Error Correction Spelling correction, a sub-problem within spell checking, is the problem of correcting a given misspelled word. One of the earliest attempts of spelling correction is based on edit distance [Damerau, 1964]. A Bayesian approach to spelling correction is the noisy channel model [Kemighan et al., 1990, Brill and Moore, 2000], which computes the correction posterior given a word prior and a corruption model. As we detail later, our approach extends this to the more modern setting which includes word contexts.

In a more modern context, there have been several approaches to detect and correct misspellings with deep neural networks. Li et al. [2018] uses a nested RNN to encode input from character-level embeddings. Li et al. [2020] uses Transformer encoder at word- and character-level. Jayanthi et al. [2020] performed a comprehensive comparison among deep models on synthetic and real misspelling correction dataset.

Compared to these approaches, we have several advantages in correcting misspellings. First, our model adopts a character-level language model and easily generalizes to rare or even unseen words, which is highly advantageous in a clinical setting where the size of the vocabulary is large. Previous models output corrections

by $|V|$-way multi-class classification, where V is the vocabulary. Also, our approach requires a small labeled misspelling dataset only for tuning hyper-parameters of the corruption model. Previous approaches require a large number of labeled misspelling examples to train the classifier and resort to synthetically generated training data, which has risk of distribution mismatch from real misspellings.

In healthcare settings, there have been several works on developing misspelling detection and correction methods [Ruch et al., 2003, Tolentino et al., 2007, Lai et al., 2015]. Fivez et al. [2017] develops a spelling correction algorithm using orthographic and phonetic edit distances and word embedding similarity. Lu et al. [2019] develops a pipeline that detects and corrects various types of spelling errors using simple rules and word embeddings. The two papers above also released real datasets of spelling errors to evaluate their performance.

Contextualized Language Models Transfer learning from pre-trained deep language models have revolutionized NLP in recent years, especially from the introduction of the Transformer architecture [Vaswani et al., 2017]. BERT [Devlin et al., 2018] uses the Transformer encoder and solves auxiliary language tasks to pre-train word embeddings. GPT [Radford et al., 2018] adopts the Transformer decoder to generate languages in an autoregressive manner. Similar to BART [Lewis et al., 2020], our model uses both Transformer encoder and decoder to take the context as input and output the correction word.

2.3 Methods

In this section, we introduce mathematical notations, the spelling correction problem, and our proposed method, CIM.

2.3.1 Problem Setup

Let $W = \{w_1, \cdots, w_{|W|}\}$ be a set of characters. This includes characters and punctuation marks used in the language of interest. The misspelled word $\mathbf{y} = [y_1, ..., y_{N_Y}] \in W^{N_Y}$, and its correction $\mathbf{x} = [x_1, ..., x_{N_X}] \in W^{N_X}$ are both sequences of characters. This character level representation of words is more suitable than subwords in our problem since the typo and the correct words substantially differ when tokenized into subwords.

Next, we define the context. We denote the vocabulary, the set of subwords used by a tokenizer, as $V = \{v_1, \cdots, v_{|V|}\}$. The context $\mathbf{c} = [\mathbf{c}_{\text{left}}, \mathbf{c}_{\text{right}}] \in V^L$, where $\mathbf{c}_{\text{left}} = [c_{-L_{\text{left}}}, ..., c_{-1}]$ and $\mathbf{c}_{\text{right}} = [c_1, ..., c_{L_{\text{right}}}]$ are part of the text occurring before and after the typo word. The overall length of the context $L = L_{\text{left}} + L_{\text{right}}$ is typically constrained by language models such as BERT.

Spelling correction, the task of finding the correction given the misspelled word and the context, can be written as the following *probabilistic inference task*:

$$\mathbf{x} = \underset{\mathbf{x} \in W^{\leq N}}{\arg\max}\, p(\mathbf{x}|\mathbf{y}, \mathbf{c}), \tag{1}$$

where N is the maximum length of the correction. An illustrated example of misspelling correction problem with the notation is shown in Figure 1.

2.3.2 Conditional Independence Model (CIM)

This subsection describes our method of spelling correction. Here, we make an assumption on the generative process of the misspelling that it only depends on the correct word, not the context. Hence, the typo word is independent to the context given the correct word: $\mathbf{y} \perp\!\!\!\perp \mathbf{c} \,|\, \mathbf{x}$. This is a reasonable assumption given that

○ **Input text:** "... with hazy ground-glass opacity in the lower lobes of the **lugns** . There is a small amount of perihepatic ascites. The evaluation of the abdomen ..."
○ **Correction**: "**lungs**"
○ **Vectors:**
- Misspelled word $\mathbf{y} = [\mathrm{l}, \mathrm{u}, \mathrm{g}, \mathrm{n}, \mathrm{s}]$
- Context $\mathbf{c} = [\mathbf{c}_{\text{left}}, \mathbf{c}_{\text{right}}]$
  - $\mathbf{c}_{\text{left}} = [..., \text{with}, \text{hazy}, ..., \text{lobes}, \text{of}, \text{the}]$
  - $\mathbf{c}_{\text{right}} = [\text{“.”}, \text{there}, \text{is}, ..., \text{the}, \text{abdomen}, ...]$
- Correction $\mathbf{x} = [\mathrm{l}, \mathrm{u}, \mathrm{n}, \mathrm{g}, \mathrm{s}]$

Figure 1: An example of misspelling correction problem

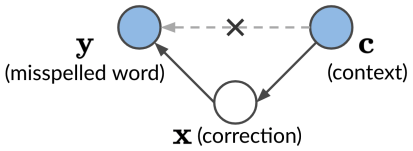


Figure 2: Graphical model of our conditional independence model. The context and the typo are observed and the correct word is unobserved.

the most cause of misspellings (homophones, typographical errors, mispronunciations) are independent to surrounding words. Please see Figure 2 for the model.

With this assumption, we can express the MAP estimator of the correction as follows:

$$\begin{aligned}\mathbf{x} &= \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}, \mathbf{c}) = \arg\max_{\mathbf{x}} p(\mathbf{x}, \mathbf{y}, \mathbf{c}) \\ &= \arg\max_{\mathbf{x}} p(\mathbf{c}) p(\mathbf{x}|\mathbf{c}) p(\mathbf{y}|\mathbf{x}) \\ &= \arg\max_{\mathbf{x}} \{\log p(\mathbf{x}|\mathbf{c}) + \log p(\mathbf{y}|\mathbf{x})\}\end{aligned} \tag{2}$$

Note that the proposed model is similar to the noisy channel model, but we include the word context which in turn entails our specific conditional independence assumption.

The first term is the language model which is the probability distribution of the correct word given the context. We model this as a transformer encoder-decoder architecture [Lewis et al., 2020], where the encoder is bidirectional same as BERT, and the decoder outputs sequence of characters in an auto-regressive, or left-to-right, manner.

The second term is the corruption model, the probability model of the typo word given the correct word. This may take into account delicate mechanism such as proximity in keyboard layout or be learned if a large dataset of misspelling is available. We adopt a simple approach that the probability is proportional to the exponential of the character edit distance between the correct word and typo word:

$$\log p(\mathbf{y}|\mathbf{x}) = -C d_{\text{ED}}(\mathbf{x}, \mathbf{y}), \tag{3}$$

where $d_{\text{ED}}(\cdot, \cdot)$ is the Damerau-Levenshtein edit distance, and C is a hyper-parameter that balances between the language model and the corruption model. We chose a simple corruption model for two reasons: to demonstrate the efficacy of the overall approach even with a simple baseline, but also that it allows us to setup a correction system with little or no data, since more complex corruption models require a training dataset of "typical misspellings" which might not always be available.

Thus since the corruption model above does not require any training, our method only requires training the character-level language model of the correct word, which can be performed with a large clinical corpus and does not require a dataset of misspellings. We further note that our approach is modular, and allows for

○ **Input text:** "... in the lower lobes of the **lugns** . There is a small amount of perihepatic ..."
○ **Misspelled word**: "**lugns**"

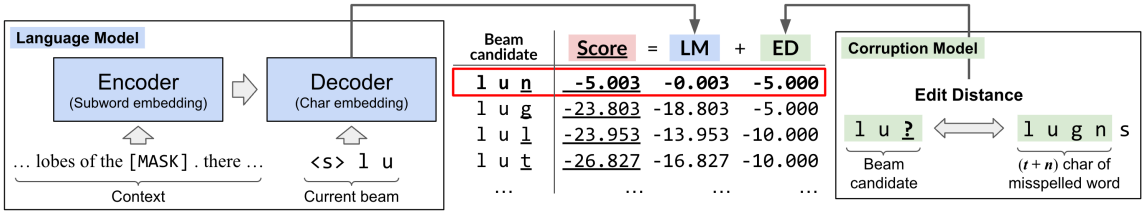


Figure 3: Beam search of CIM on the Example 1 at time step $t = 3$. The beam candidates are ranked by the sum of the language model score (LM) and the corruption model score (ED). The hyper-parameters of the corruption model are $C = 5.0$ and $n = 1$. The beam width is chosen to $B = 1$ for clear visualization.

incorporating any future advances in either language models, or corruption models (for instance if large-scale misspellings datasets become available).

2.3.3 Beam Search with the Two Model Components

After training the language model, we combine the two models at decoding phase to perform misspelling correction. At the time step t of beam search, the model outputs candidates by expanding candidates from the previous time step and sorting them by intermediate scores:

$$\mathcal{B}_t = \underset{\mathbf{x}_{:t} \in (\mathcal{B}_{t-1} \times W)}{\arg\max_B} \left\{ \log p(\mathbf{x}_{:t}|\mathbf{c}) + \log p(\mathbf{y}|\mathbf{x}_{:t}) \right\}, \tag{4}$$

where $\mathcal{B}_t$ is the set of candidates up to length t, B is the beam width, and $\arg\max_k$ is top-k argmax.

The first term, the language model score can be obtained naturally. The second term, the corruption model score is computed by the edit distance of the partial output and the first $t + n$ characters of the typo word:

$$\log p(\mathbf{y}|\mathbf{x}_{:t}) = -C d_{\text{ED}}(\mathbf{x}_{:t}, \mathbf{y}_{:t+n}). \tag{5}$$

This prevents the edit distance score from advantaging the characters of the typo word far behind the t-th position. The hyper-parameter n implicitly assumes how many characters can be inserted, at most, to corrupt a word. Also, we restrict the possible set of the correct words to be the predefined dictionary. Combining the two scores and the dictionary constraint, the beam search step of our method becomes as follows:

$$\mathcal{B}_t = \underset{\mathbf{x}_{:t} \in (\mathcal{B}_{t-1} \times W) \cap D_t}{\arg\max_B} \left\{ \sum_{i=1}^{t} \log p(\mathbf{x}_i|\mathbf{x}_{<i}, \mathbf{c}) - C d_{\text{ED}}(\mathbf{x}_{:t}, \mathbf{y}_{:t+n}) \right\}, \tag{6}$$

where D_t is the set of length-t substrings of the dictionary words. After generating the candidate words, we choose the best candidate by the scores of the candidates normalized by their lengths, following the practice of beam search. Please see Figure 3 for the visualization of the beam search of CIM.

2.4 Experiments

2.4.1 Datasets

There are few publicly available clinical misspelling datasets annotated by human experts. Here we describe two datasets to tune and evaluate our method for misspelling correction.

Method	Acc(%)
Fivez et al. [2017]	92.10
CIM-Base (B=30)	95.07
CIM-Base (B=300)	95.99
CIM-Large (B=30)	95.65
CIM-Large (B=300)	**96.56**

Table 1: Accuracy results for the MIMIC-III misspelling datasets (B is the width of beam search).

Method	Acc(%)
Lu et al. [2019]	54.70
CIM-Base (B=30)	67.07
CIM-Base (B=300)	**69.51**
CIM-Large (B=30)	65.68
CIM-Large (B=300)	67.60

Table 2: Accuracy results for the CSpell test set (B is the width of beam search).

Hparam	Models	
	CIM-Base	CIM-Large
Early stop	475k steps	300k steps
C	5.0	5.0
n	1	1

Table 3: Hyper-parameters of CIM tuned on CSpell training dataset.

MIMIC-III Misspelling Dataset Fivez et al. [2017] released a manually annotated dataset of clinical misspellings from the clinical notes in the MIMIC-III database. This single-set dataset contains 873 instances of 357 non-word misspellings. After a carefully review of the examples by a medical doctor, we found that the labels of 30 examples are incorrect, and updated the dataset.

CSpell Spelling Error Dataset Lu et al. [2019] released a dataset of various types of spelling errors. The dataset is collected from consumer health questions to their QA system and covers a wide range of errors other than misspellings, such as to-merge and to-split errors. Since we focus on evaluating misspelling correction, we extract single-word misspellings that contain only alphabets. Their spelling checking software, CSpell, both detects and corrects spelling errors. To make a fair comparison with CSpell, we further excluded examples from the test set that are not detected by any of the detection modules of CSpell. As results, 409 and 574 examples are chosen from the training set and the test set, respectively.

2.4.2 Implementation

The implementation of the language model of CIM is based on the BART implementation of Hugging Face's Transformers [Wolf et al., 2020]. The encoder part is same as BERT [Devlin et al., 2018] and initialized with BlueBERT [Peng et al., 2019], a clinical version of BERT. We use the same number of Transformer decoder layers as the encoder. We denote the model with BlueBERT-Large (24-layer) as CIM-Large and the model with BlueBERT-Base (12-layer) as CIM-Base. As a result, CIM-Base and CIM-Large have 132M and 403M parameters, respectively. The reference dictionary is built by combining an English dictionary DWYL [2020] and a medical lexicon, the `LRWD` and `prevariants` tables of Unified Medical Language System (UMLS).

We trained the language model of CIM on the clinical notes of the MIMIC-III dataset. Both CIM-Base and CIM-Large are trained for 500k iterations with batch size 256 on 4×NVIDIA A100 GPU 40GB. We follow the optimizer and learning rate schedule of BERT, except we reduced the learning rate of the encoder since the encoder is initialized with BlueBERT while the decoder is randomly initialized. Hyper-parameter search is performed over various values of C, n, and the training steps to maximize the correction accuracy on the CSpell training set. The beam width is fixed to B=30 during the tuning for faster search.

2.5 Results and Analysis

In this section, we report the results of CIM on the two real datasets of clinical misspelling, and perform additional analyses on CIM.

[Input]
Typo / Correct : noited / noted
Context: "to the previous tracing occasional atrial ectopy is **noited** . Othuerwise, no significant change. Left bundle-branch block"
[Output]

Large/300k/B300/ED1(1)	Score	LM	ED
noted	-1.0378	-0.2045	-0.8333
noticed	-1.9934	-0.7434	-1.2500
noised	-2.6872	-1.9729	-0.7143
indicated	-3.0067	-1.0067	-2.0000
notified	-3.1666	-1.5000	-1.6667
continued	-3.2132	-0.7132	-2.5000
identified	-3.2166	-0.4893	-2.7273
normalized	-3.2963	-1.0236	-2.2727
confirmed	-3.3180	-0.8180	-2.5000
voiced	-3.3584	-1.9298	-1.4286

(a) MIMIC-III example (success)

[Input]
Typo / Correct : cronic / chronic
Context: "ClinicalTrials.gov - General Complaint. I HAVE **cronic** PAIN FROM SEVERE SHINGLES THE BLISTERS ARE GONE BUT THE PAIN IS SEVERE WHAT SHOULD"
[Output]

Base/475k/B300/ED1(1)	Score	LM	ED
chronic	-1.1245	-0.4995	-0.6250
chronics	-2.8642	-1.7531	-1.1111
chronical	-2.9247	-1.4247	-1.5000
chronicle	-2.9929	-1.4929	-1.5000
chronically	-3.2452	-1.1618	-2.0833
chronicity	-3.2661	-1.4479	-1.8182
cranial	-3.2816	-1.4066	-1.8750
chronica	-3.3225	-2.2114	-1.1111
atonic	-3.3835	-1.9550	-1.4286
chronicles	-3.4004	-1.5823	-1.8182

(b) CSpell Test example (success)

[Input]
Typo / Correct : allopatic / allopathic
Context: "(antifungal which cure it. but I am worried of taking **allopatic** medicines. please tell me what primitive or traditional methods do"
[Output]

Base/475k/B300/ED1(2)	Score	LM	ED
alloplastic	-3.1710	-2.3376	-0.8333
allopathic	-3.2432	-2.7886	-0.4545
alkalotic	-3.3652	-1.3652	-2.0000
elliptic	-3.4361	-1.7694	-1.6667
hallucinating	-3.4598	-0.9598	-2.5000
alternative	-3.4706	-0.9706	-2.5000
myopathic	-3.4711	-1.4711	-2.0000
prophylactic	-3.5400	-0.8477	-2.6923
allopurinol	-3.5600	-1.4767	-2.0833
allegation	-3.5933	-1.7751	-1.8182

(c) CSpell Test example (failure)

Figure 4: Beam search decoding examples. For each example, we display the top 10 beam candidates. The column next to the candidate (`Score`) shows the final beam score for each candidate.

2.5.1 Results on Clinical Misspelling Datasets

We report our results and those of baselines for the two real misspelling datasets in Table 1 and Table 2. The hyper-parameters chosen for Beam search decoding are shown in Table 3. For both of the datasets, our method outperforms the dataset baselines by large margin.

In all settings, the accuracy increases as the beam width increases. One interesting observation is that CIM-Large performs better than CIM-Base on the MIMIC-III dataset but worse on the CSpell dataset. We think that this is because CIM-Large overfits the word distribution of the MIMIC-III notes, which is different from the health consumer questions.

Figure 4 shows some beam search examples of CIM. The candidates are generated and ranked by their scores (`Score`) consisting of the language model score (`LM`) and corruption model score (`ED`). Figure 4b shows an easy case where both modules give the highest scores to the correct word. In a more challenging example such as Figure 4a, there is a candidate word ("noised") that has a higher corruption model score than the correct word ("noted"), but the language model gives a much higher score to the correct word. Figure 4c shows a failure case of CIM, where the language model gave a low score to the correct word.

2.5.2 Analysis

Ablation Study of Model Components To see the effect of each component of our misspelling correction model, we conducted an ablation study of the corruption model and the reference dictionary. For each configuration, the hyper-parameters are tuned independently, and the evaluation was performed with the beam width $B = 300$.

The first four rows of Table 4 shows the results of the ablation study. The most noticeable result is that the corruption model contributes significantly to the model's performance. This is predictable because, with only the language model, the output would be any word that fits into the context, regardless of the misspelled word. Another observation is that dictionary matching contributes to the model only when both the language model and the corruption model are used. This is because the reference dictionary is unnecessary for the "LM only" setting since the language model is trained to produce dictionary words. However, when combined with the corruption model, the chance to output non-dictionary words increases, so the reference dictionary helps our model.

Effect of Hyper-parameters To see the effects of the hyper-parameters, we evaluate our model with various values of C and n. Table 5 shows the results of CIM-Base on the CSpell test set with various C and n. We

Method \ Dataset	MIMIC-III		CSpell Test	
	Base	Large	Base	Large
LM only	37.57	37.34	20.91	20.21
LM + Dict	37.57	37.34	20.91	20.21
LM + ED	93.24	92.67	66.72	66.03
LM + ED + Dict	**95.99**	**96.56**	**69.51**	67.60
Unsupervised	95.07	95.42	68.29	**68.99**

Table 4: Accuracy results of ablation study and unsupervised setting. LM: language model, ED: corruption model, Dict: dictionary matching, Unsupervised: tuning on the synthetic dataset (B=300)

$C \setminus n$	0	1	2	∞
2.0	64.29	64.63	63.94	61.32
5.0	66.03	**67.07**	66.20	63.94
10.0	56.97	58.01	57.32	52.96
20.0	46.34	49.83	48.78	44.25

Table 5: Accuracy results on the CSpell test set with different values of C and n (CIM-Base with B=30).

Dataset \ Semantic Type		Substance	Disease	Symptom	Other	(Total)
MIMIC-III	Fivez et al. [2017]	91.24	89.87	94.83	91.52	92.10
	CIM	**95.62**	**96.20**	**98.28**	**96.34**	**96.56**
	Count	137	79	174	500	873
CSpell Test	Lu et al. [2019]	54.31	70.59	57.89	48.77	54.70
	CIM	**68.10**	**72.94**	**77.63**	**66.05**	**69.51**
	Count	116	85	76	324	574

Table 6: Results of subgroup analysis by UMLS Semantic Types

choose $B = 30$ for the beam search. We can see that the best accuracy is achieved when the hyper-parameters are tuned to the CSpell train set ($C = 5.0$ and $n = 1$), which suggests the CSpell test distribution align with the train set. Also, the accuracy decreases as the values of C and n move away from their optimal values. This is predictable since these hyper-parameters balances the model's preference for different candidates. In other words, increasing C makes the model prefer candidates similar to the misspelled word, and decreasing it makes the model prefer candidates fitting the context better.

Subgroup Analysis by Semantic Types To see the effectiveness of CIM on different types of words, we computed the subgroup accuracy according to UMLS Semantic Type. We first chose three subtrees in the UMLS Semantic Type hierarchy for three subgroups of words, namely "substance", "disease", and "symptom". Then, for each example, we query the correction word to UMLS for semantic types and include the example into a subgroup if any semantic types of the correction word fall into the subgroup. We also grouped examples that did not belong to any of the subgroups into "other" subgroup. Note that an example can belong to more than one subgroup.

Table 6 shows the results of the subgroup analysis. In all subgroups of both datasets, CIM consistently outperforms baselines. In the CSpell dataset, CIM performs better on "Disease" and "symptom" subgroups than others, and similarly in the MIMIC-III test set, CIM shows the best accuracy on "Symptom" subgroup and most significant improvement over baseline on "Disease" subgroup.

Results in Unsupervised Setting While we used a small set of real misspellings for the hyper-parameter search, we can tune the hyper-parameters even in an unsupervised setting with a synthetic dataset of

misspellings. From the MIMIC-III clinical notes, we randomly choose words that are in our reference dictionary and corrupt them with random operations, which results in 10k examples of synthetic misspellings. To corrupt words, up to two operations of character addition, deletion, substitution, or transposition can be applied.

We performed the hyper-parameter search on this synthetic dataset, as we did with the CSpell training set. The last row of Table 4 shows the results, and the test accuracy under the fully unsupervised setting is comparable to the supervised setting.

2.6 Conclusion

The main contribution of the paper is to present a well-formalized spelling correction method that combines a deep neural language model and the corruption model. Our experiments show that the method outperforms the baseline methods, including an off-the-shelf software. Although the main concern of the paper is healthcare text, our method can be applied to other areas with specialized lexicons or general misspelling correction. Two important directions for improvement are to develop an improved corruption model and to extend the model to deal with multiple-word spelling errors.

3 Structured Attention for Compositional Generalization

This section is based on Kim et al. [2021].

3.1 Introduction

Compositional generalization is the ability of a system to systematically generalize to a new data distribution by combining known components or primitives. For example, assume a system has learned the meaning of "jump" and that "jump twice" means that the action "jump" has to be repeated two times. Upon learning the meaning of the action "jax", it should be able to infer what "jax twice" means. Although modern neural architectures are pushing the state of the art in many complex natural language tasks, these models still struggle with compositional generalization Hupkes et al. [2020].

In order to advance research in this important direction, in this paper we present two main contributions [1]. First, we present a binary classification dataset which is hard in a compositional way. This allows for studying the compositional generalization ability of a larger range of models than sequence generation tasks, since the task only requires an encoder, and not a decoder. Specifically, we present a methodology to convert an existing semantic parsing dataset, CFQ Keysers et al. [2019], into a binary classification dataset that is also compositionally hard.

Our second and main contribution is showing that a transformer-based model can better generalize compositionally if we provide hints on the structure of the input. Specifically, we do so by modifying the attention mask used by the model. This is an interesting result, as (except for two additions, which we elaborate on in Section 3.4) attention masks do not "add" any attention capabilities to the model. Instead, it seems that it is the removal of certain attention pairs that makes the difference. This suggests that vanilla Transformer is having a hard time suppressing non-compositional attention.

3.2 Background

This section overviews existing work on compositional generalization and then some background on the Transformer models used in this paper.

Compositional Generalization Compositional generalization can manifest in different ways Hupkes et al. [2020] such as *systematicity* (recombination of known parts and rules) or *productivity* (extrapolation to longer sequences than those seen during training), among others. Early work focused on showing how different deep learning models do not generalize compositionally Liška et al. [2018], and datasets such as SCAN Lake and Baroni [2018] or CFQ Keysers et al. [2019] were proposed to show these effects.

Work toward improving compositional generalization has proposed ideas such as Syntactic attention Russin et al. [2019], increased pre-training Furrer et al. [2020], data augmentation Andreas [2019], or general purpose sequential models such as *Neural Turing Machines* or *Differential Neural Computers* Graves et al. [2016].

Extended Transformer Construction For our experimental evaluation we use the ETC Ainslie et al. [2020] Transformer model. ETC extends the standard Transformer model in 3 key ways: (1) it uses a global-local attention mechanism to scale to long inputs, (2) it uses relative attention Shaw et al. [2018] and flexible masking and (3) it uses a new pre-training loss based on Contrastive Predictive Coding (CPC) Oord et al. [2018]. The last two extensions allow it to handle structured inputs containing, for example, hierarchical structure. In this work, we rely on (2) to annotate the structure of the input.

[1] `http://goo.gle/compositional-classification`

• **Question (CFQ input)**

```
Did M0 ' s writer , editor , director , and cinematographer found M1 and found M2
```

• **Positive query (CFQ output)**

```
SELECT count ( * ) WHERE {
 ?x0 film.cinematographer.film M0 .
 ?x0 film.director.film M0 .
 ?x0 film.editor.film M0 .
 ?x0 film.writer.film M0 .
 ?x0 organizations_founded~ M1 .
 ?x0 organizations_founded~ M2
}
→ Label: 1 (same)
```

• **Model negative query**

```
SELECT count ( * ) WHERE {
 ?x0 film.cinematographer.film M0 .
 ?x0 film.director.film M0 .

 ?x0 film.writer.film M0 .
 ?x0 organizations_founded~ M1 .
 ?x0 organizations_founded~ M2
}
→ Label: 0 (different)
```

• **Random negative query**

```
SELECT DISTINCT ?x0 WHERE {
 ?x0 a film.editor .
 ?x0 film.writer.film M1 .
 ?x0 film.writer.film M2 .
 ?x0 film.writer.film M3 .
 ?x0 ~.person.nationality m_0d060g
}

→ Label: 0 (different)
```

Figure 5: Examples of the CFQ classification dataset. Each query pairs with the question to form an instance. Note the model negative resembles the positive, while the random negative query differs considerably.

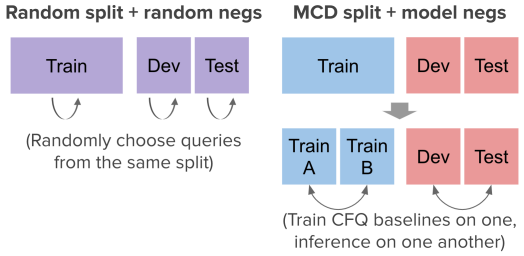


Figure 6: Negative example strategies. Different colors indicate different compound distributions.

3.3 The CFQ Classification Dataset

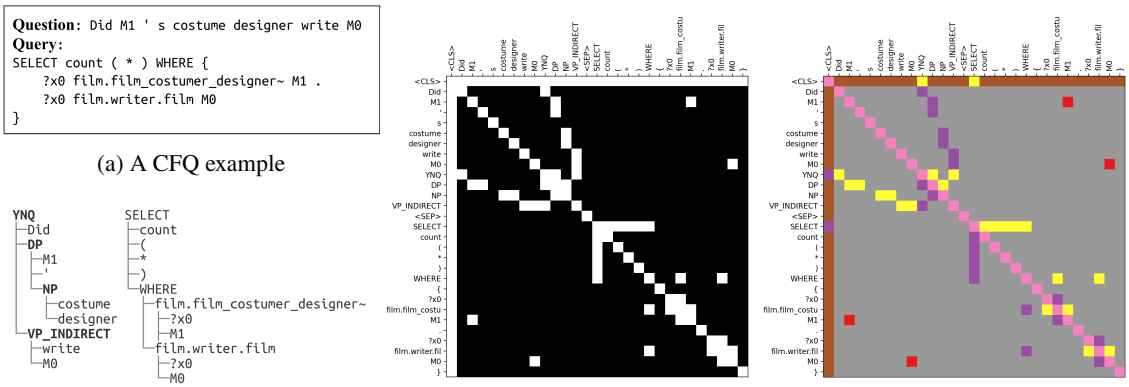
The Compositional Freebase Questions (CFQ) dataset [Keysers et al., 2019] is an NLU dataset to measure the compositional capability of a learner. It is designed around the task of *translating* a natural language question into a SPARQL query. The dataset has been automatically generated by a grammar and contains 239,357 sentence/query pairs. An example is shown in Figure 7a.

As shown in the original work of Keysers et al. [2019] in order to properly measure the compositional generalization ability of a model, the train and test sets should be split with similar distributions of tokens (*atoms*), but different distributions of their compositions (the *compounds*). In the CFQ dataset, to ensure this, two divergences, namely *atom divergence* and *compound divergence*, between the train and dev/test set are measured while constructing the splits. As a result, carefully selected splits called *maximum compound divergence (MCD)* splits are hard for standard neural networks (they perform well in the train set, but poorly in the test set), while the random splits are easier.

We convert the CFQ dataset into a dataset with a binary classification task. In this new dataset, the input is a question and a SPARQL query, and the task is to determine whether these two sequences have the same meaning or not. Two considerations must be made to ensure the resulting dataset requires compositional generalization:

Negative Example Strategies: Positive instances of the binary classification task can be obtained directly from the original dataset, but to obtain negatives, we use either of two strategies:

- Random negatives: We pair each question with a randomly chosen query.



(a) A CFQ example

(b) Parse trees of the CFQ example

(c) Hard mask

(d) Soft mask

Figure 7: Structure annotations for a CFQ example. We extract the hierarchical structure of the question and query of CFQ examples and use them to mask attention (hard mask) and/or provide relative attention labels (soft). Different colors indicate different relative attention labels.

- Model negatives: Using baseline models (LSTM Hochreiter and Schmidhuber [1997], Transformer Vaswani et al. [2017], and Universal Transformer Dehghani et al. [2018]) trained on the original CFQ dataset, we get top-k query predictions for each question. After filtering syntactically invalid queries and duplicates, we can get hard examples for classification from their incorrect predictions.

Model negatives are important, as otherwise, the task becomes too easy and would likely not require compositional generalization. See Figure 5 for examples of random/model negative instances.

Compound Distribution of Negative Examples: To prevent data leakage (e.g., compounds from the test set of the original CFQ dataset leaking into the training set of the classification CFQ dataset), we carefully choose the sampling set for random negatives and the train and inference set for model negatives. We generate two splits of the original CFQ dataset. Each split contains three sets with 50% data on train, 25% on dev and 25% on test. The first is a *random split* of the data, and the second (*MCD split*), maximizes the compound divergence between train and dev/test using the same method as in the original CFQ work. Then, we process the examples in each of these sets generating positive and negative examples. For random negatives, we sample negative queries for each questions from the set which the original example belongs to (train/dev/test). For model negatives, to generate negatives for the training set, we divide it into two halves, train models in one, and generate negatives with the other half. For dev/test, we train on dev and generate negatives on test, and vice versa. Figure 6 illustrates this procedure, designed to ensure there is no leakage of compounds between train and dev/test.

For both strategies, we make 1 positive and 3 negatives per original CFQ example. Also, we set aside 5% of the train set as a hold-out set to check i.i.d. generalization.

3.4 Compositional Generalization via Structure Annotation

Our hypothesis is that part of the difficulty in compositional generalization is to parse the structure of the input. To test this, we evaluate the performance of models when we provide annotations for two structural elements

Model	*Random Split & Random Neg*			*MCD Split & Model Neg*		
	Train	**Hold-out**	**Dev**	**Train**	**Hold-out**	**Dev**
LSTM	1.0000	0.9998	0.9998	1.0000	0.9972	0.8310
Transformer (2 layers)	0.9998	0.9997	0.9998	0.9988	0.9931	0.8789
Transformer (6 layers)	0.9999	1.0000	0.9999	0.9995	0.9931	0.8738

Table 7: AUC on the CFQ classification dataset generated with different methods

of the inputs: parse trees of both the natural language sentences and SPARQL queries, and *entity cross links* (linking entity mentions from the natural language side to the corresponding mentions in the SPARQL query).

The parse trees of the questions are already given in the original CFQ dataset as constituency-based parse trees. Since the trees include intermediate nodes indicating syntactic structures, we append tokens representing them at the end of each question. We created a simple parser to generate dependency-based parse trees for the SPARQL queries. We join the roots of the two trees to make a single global tree with the <CLS> token as the root.

We represent the structure of the inputs by masking attention ("hard mask") or with relative attention [Shaw et al., 2018] labels ("soft mask").

- Hard mask: We customize the binary attention mask of the original Transformer to only allow attention between tokens connected by the edges of the parse tree.
- Soft mask: For every pair of input tokens, we assign relative attention labels based on which of the following edge relationships applies: parent-to-child, child-to-parent, itself, from-or-to-root, or entity-cross-link.

Additionally, we allow attention pairs in the masks connecting the entities appearing both in the question and the queries. We call these links *entity cross links*, and they are found by simple string match (e.g. "M0"). Notice that while relative attention labels and the additional tokens to represent the constituency parse tree of the natural language add capabilities to the model, the "hard mask" structure annotations described above (which result in the larger performance gains) do not *add* any attention capabilities to the model. Instead, they simply remove non-structure attention edges. Figure 7b shows the parse trees, and Figure 7c and 7d show the masks for an example.

3.5 Experiments

We used the ETC Ainslie et al. [2020] Transformer model implementation as it allows us to provide the hard and soft masks described above in an easy way. In all experiments, we report AUC in the dev set as the evaluation metric (we did not evaluate on the test set).

The CFQ Classification Dataset We generate two classification datasets: "random split & random negatives" and "MCD split & model negatives", and evaluate LSTM and Transformer models. For both datasets, we evaluate AUC on the hold-out set (taken out of the training set as described above) to test i.i.d. generalization, and on the dev set to test compositional generalization.

As shown in Table 7, models easily generalize on the hold-out set (AUC $\geq$ 0.99). All baseline models also achieve almost 1.0 AUC in the dev set of the "random split & random negatives". However, in the "MCD split & model negatives" models cannot generalize well on the dev set, showing compositional generalization is required. Note that random guessing achieves 0.5 AUC score.

Model	Mask Type	Cross link	*MCD Split & Model Neg* Train	Hold-out	Dev
LSTM	-		1.0000	0.9972	0.8310
Transformer	-		0.9995	0.9931	0.8738
Transformer w/ structure annotations (ETC)	No	-	0.9994	0.9934	0.8868
	Hard	N	0.9999	0.9978	0.9061
		Y	1.0000	0.9992	<u>0.9656</u>
	Soft		0.9995	0.9936	0.8819
	Both		1.0000	0.9991	**0.9721**

Table 8: AUC on the CFQ classification dataset (*MCD Split & Model Neg*) with various structure annotations

Structure Annotation Table 8 compares different ablations of our structure annotation approach compared to the baseline models. The first (no masks and no cross links) just shows that adding tokens to the input to represent the constituency parsing and moving to ETC only provide small gains (from 0.8738 to 0.8868 AUC). Adding a hard mask already helps the model (0.9061 AUC), and adding cross links on top of that achieves very significant gains (0.9656 AUC). Finally, soft masks by themselves do not seem to help, but a combination of soft and hard masks achieves our best result of 0.9721 AUC.

The interesting result here is that adding the hard mask with entity cross links *only removes* potential attention pairs, so it does not increase model capacity in any way. In other words, the underlying transformer model is in principle able to generalize compositionally to some extent but seems to struggle in suppressing non-compositional attention.

3.6 Conclusions

The main contribution of this paper is to show that providing structure annotations in the form of attention masks significantly helps Transformer models generalize compositionally. This is interesting for two main reasons: first, it shows that neural network models do have the innate ability to generalize compositionally to some extent, but need some guidance to do so (e.g., by providing attention masks as in our work). This reinforces previous work showing that LSTMs also can, in principle, generalize compositionally, but they just do so with very low probability Liška et al. [2018]. The second reason is that structure annotations, which we provided manually, could be generated by another model in future work. We also presented a procedure for generating classification datasets that require some degree of compositional generalization starting from sequence generation datasets.

4 Multimodal Modeling of Clinical Event Timelines

This section is based on Frattallone-Llado et al. [2024].

4.1 Introduction

Temporal data mining involves the extraction of temporal information from different sources and modalities of data, and it has broad application in fields such as law, finance, and healthcare. For instance, in criminal recidivism prediction, the event timeline for a defendant could be extracted from both texts in probation office documents and tables in psychiatric health records Cheng et al. [2009]; In stock price movement and volatility prediction, financial time series could be extracted from financial news, daily stock market price tables, and verbal and vocal cues in earning calls Ang and Lim [2022]. In clinical risk prediction, patient timeline could be extracted from electronic health records with both unstructured clinical notes and structured tabular data Tayefi and et al. [2021]. This paper focuses on providing a multimodal extraction system and a benchmark dataset for clinical timelines.

Precise clinical event timelines are crucial for prognosis and prediction tasks. These forecasting tasks have been studied with varied prediction times and unstructured data sources Seinen et al. [2022]. Discharge summaries provide the most complete information. In the 2012 Informatics for Integrating Biology and Bedside (i2b2) Challenge Sun et al. [2013b], 310 discharge summaries were annotated with temporal information, including clinical events, temporal expressions, and their temporal relations.

This approach yields a relative timeline of clinical events, rather than absolute, leaving many events without the precise timing needed for forecasting tasks. To achieve a more complete event timeline, i2b2 events were annotated with absolute time values Leeuwenberg and Moens [2020], by bounding the events with closed intervals in calendar times and temporal uncertainties on the bounds. Their annotation procedure was unimodal, as annotators had access only to the discharge summary text.

Meanwhile, there is a consensus that the integration of structured and unstructured data has a significant impact on constructing models and predicting target variables Seinen et al. [2022]. In this project, we take advantage of the combination of unstructured and structured data in a multimodal approach, which has proven beneficial in other applications Liu et al. [2022], Moldwin et al. [2021]. Our work adopts a version of the probabilistic bounds described and applies them to discharge summaries from the i2b2 dataset in order to generate absolute inpatient event timelines. We introduce the following: a visualization and annotation tool, an annotation process with a three-pass system, two types of annotations to better represent the nature of clinical events, and the multimodal annotation approach. By combining the information from unstructured and structured data, this multimodal approach should yield a more precise (*i.e.*, less uncertain) timeline for inpatient events.

Our multimodal approach contributes the following: (i) we introduce absolute timeline intervals without assumption of independence of endpoint uncertainties, (ii) we find that multimodal annotations lead to more precise timelines than the unimodal annotations. (iii) we verify the annotation quality by mapping our annotations to temporal relations, where our relations compare favorably against benchmark annotations (i2b2), and (iv) we demonstrate that a fine-tuned multimodal encoder (BERT) architecture outperforms fine-tuned unimodal encoder and off-the-shelf generative encoder-decoder architectures (Llama-2). Overall, we show the importance of annotation from multimodal data sources, both in the annotation process and for machine learning predictive performance.

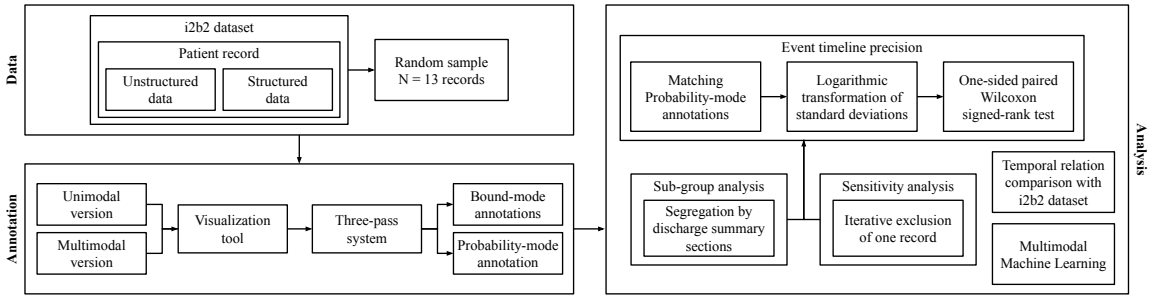


Figure 8: Flowchart of our method.

4.2 Methods

We use the i2b2 dataset Uzuner et al. [2011], a compendium of de-identified electronic health records from Partner Healthcare and Beth Israel Deaconess Medical Center, containing discharge summaries with annotated clinical time/events. We located i2b2 patient's records in MIMIC-III with matching discharge summaries, allowing us to collect both structured and unstructured records. Thirteen records were randomly selected from the training set of the 2012 i2b2 temporal relations challenge data Sun et al. [2013b]. For each of these records, a human annotator with medical experience identified clinical events in the discharge summary and timestamped their endpoints on a timeline.

In our study, an annotation comprises a contiguous, highlighted text, representing a clinical event, and a time interval. Based on the use of the structured and unstructured data, two annotation versions were generated: multimodal and unimodal. In the unimodal version, the annotator only had access to the discharge summary and the admission and discharge times. In the multimodal version, the annotator also had access to the full structured data. Out of the total thirteen records, five were annotated using both unimodal and multimodal versions, while the remaining records were annotated from only multimodal. Upon acceptance, the annotation tool, the annotation files, and the analyses will be made public. A flowchart representation of the process describes the steps of our annotation and analysis (Fig. 8).

4.2.1 Annotation Tool

The R Shiny annotation tool displays all the unstructured and structured data for any given record (Fig. 9). The structured data is displayed on a graph, where the x-axis contains absolute time values and the y-axis contains event identifiers. The user annotates time intervals by clicking and dragging on the graph (blue overlay). Structured data events contained within this overlay are displayed in a table that the user can search and select to be relevant. On the unstructured data, or discharge summary, section, the user selects the relevant span for annotation by highlighting the text. The polarity of negative events is specified with a checkbox. Both types of annotation (Bounds- and Probability- mode) require the user to generate an overlay on the graphical timeline. In Bounds-mode annotation, either the lower or upper bound can be omitted by unchecking a box, which represents *indefiniteness*. In Probability-mode annotation, users also select standard deviations to represent their uncertainty about the lower bound, upper bound, and duration of the event. The choices are pre-defined as follows: 1, 3, or 10 seconds; 1, 3, 10, or 30 minutes; 1, 2, 4, 6, 12, or 24 hours; 2 days; 1 or 2 weeks; 1 month; 1, 10, or 100 years.

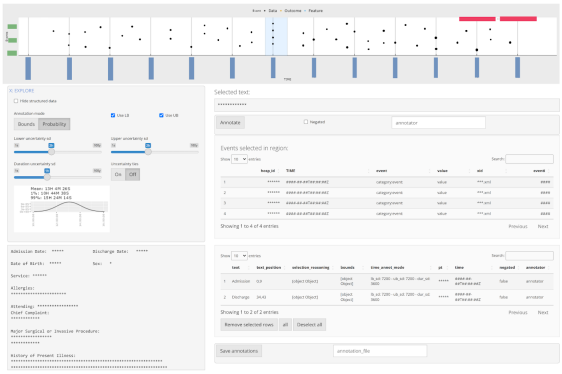


Figure 9: The web-based annotation tool.

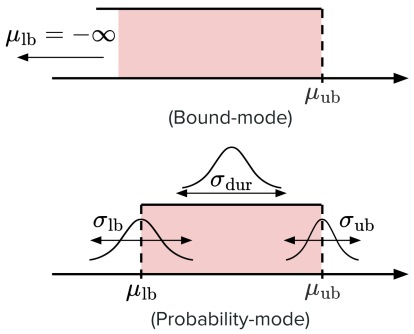


Figure 10: Two annotation modes

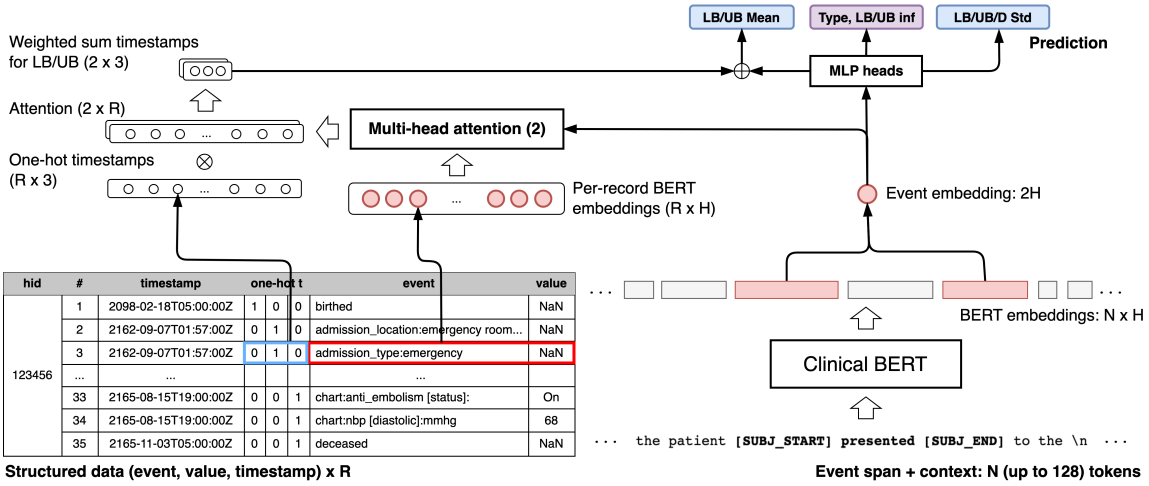
4.2.2 Annotation Process

The annotation process consisted of a three-pass system defined as follows: (1) the annotator peruses the document in its entirety to become familiarized with its content, (2) the annotator makes annotations at the paragraph level, setting lower and upper bounds that apply to all the events in that paragraph, and (3) the annotator makes annotations on particular events, which could represent words or phrases. Events are selected according to the i2b2 annotation guidelines Sun et al. [2013a]. For simplicity, the one event attribute recorded is the polarity. As stated previously, two types of annotations are available: Bounds-mode and Probability-mode. In Bounds-mode annotation, lower and upper bounds are defined, with the expectation that the event(s) occurs at some point between them, and no information is specified about event duration or timing uncertainty. One can omit either bound when the value is unknowable, as in the lower bound for certain conditions from the past medical history. In this case, the bound defaults to negative or positive infinity, as appropriate. Bounds-mode annotation is predominantly used in the second pass, because each paragraph may contain various events with different timing attributes. In Probability-mode annotation, the selected lower and upper bounds represent the mean values of two distributions. The mean value of the event's duration is calculated as their difference. All three distributions are assumed to be normal and the annotator must select a standard deviation for each one. These standard deviations represent the annotator's level of uncertainty about the event's timing and will serve as a surrogate of timeline precision for data analysis. Probability-mode annotation is used predominantly in the third pass, where the duration and approximate bounds of particular events may be reasonably determined.

4.2.3 Statistical Methods

The annotation process above was performed on the thirteen randomly-selected records from the i2b2 training dataset Sun et al. [2013b]. Five records are annotated from both unimodal and multimodal data by performing the process twice for each of these records, while the rest are annotated solely from the multimodal version of the data. In total, eighteen annotation files were generated, comprising five unimodal and thirteen multimodal versions. These files contain 4884 annotations in total, of which 1156 were in Bounds-mode and 3728 were in Probability-mode.

We performed two descriptive analyses. For the first analysis, we focus on the ten annotation files originated from both unimodal and multimodal data to study and compare the annotation practices across both versions. We defined the main effect as the precision of the event timelines (multimodal vs unimodal), *i.e.*, the difference



hid	#	timestamp	one-hot t			event	value
123456	1	2098-02-18T05:00:00Z	1	0	0	birthed	NaN
	2	2162-09-07T01:57:00Z	0	1	0	admission_location:emergency room...	NaN
	3	2162-09-07T01:57:00Z	0	1	0	admission_type:emergency	NaN
	...	...				...	
	33	2165-08-15T19:00:00Z	0	0	1	chart:anti_embolism [status]:	On
	34	2165-08-15T19:00:00Z	0	0	1	chart:nbp [diastolic]:mmhg	68
	35	2165-11-03T05:00:00Z	0	0	1	deceased	NaN

Figure 11: Multimodal BERT model for clinical event timeline prediction

in uncertainty of events is obtained by comparing the standard deviation for the lower bound, upper bound, and duration for multimodal and unimodal annotations. To compare these event timelines, we selected exact matches from the Probability-mode annotations, and we performed a one-sided paired Wilcoxon signed-rank test on the logarithmic transformation of the standard deviations. The resulting estimated differences between the two versions correspond to a scaling factor, which reflects the degree of increase or decrease in uncertainty, which we call *precision factor*. Confidence intervals were calculated using the bootstrap. To verify the robustness of the results, a sensitivity analysis was performed by re-running the test while iteratively excluding one record pair.

For the second analyses, we sought to assess the compatibility between our annotations and the i2b2 dataset. We aligned the events in the thirteen multimodal annotation files to the events in the corresponding i2b2 dataset. Since the i2b2 dataset provides only temporal relations between two events – "BEFORE", "AFTER", or "OVERLAP" – while ours provides the absolute time of events, we extracted those i2b2 temporal relations where both text endpoints could be matched with our annotated events, and compared to the temporal relations computed from our annotations. We used character-level intersection over union to match the text span of the i2b2 events and ours. We restricted our events to be aligned to be Probability-mode and calculated a pair of z-scores: one indicating the likelihood that one event precedes the other and another for the opposite order. If exactly one z-score exceeds a predetermined threshold, we determined that one event precedes the other, otherwise they overlap. After matching the temporal relations between our dataset and the i2b2 dataset, we report the F1 score, inter-annotator agreement, and accuracy of the types of relations.

4.2.4 Multimodal Learning

To test the utility of multimodal data over single-modal data, we study a reduced version of our absolute timeline prediction. The task involves binary classifications of whether the lower bound (LB) and upper bound (UB) are finite, identifying the type of annotation (Bounds- or Probability-mode), and multi-class classifications of the bounds and the standard deviations of LB, UB, and duration. Each of the bound and standard deviation classifications consists of three classes defined as follows. Bounds are classified relative to admission time, with thresholds at admission and 24 hours post-admission. Standard deviations for LB

and UB are grouped into three classes: under 2 hours, 4 hours to 1 day, and over 2 days. Duration standard deviations follow a similar pattern, classified as under 1 hour, 2 hours to 1 day, and over 2 days.

The experiment on the classification version of our dataset employed two BERT-based models. The first model, named Unimodal BERT, is a span classification model Zhong and Chen [2021] where the contextualized BERT embeddings of the annotated text span from clinical notes along with its context are fed to a feedforward network for classification. The second model, named Multimodal BERT, also incorporates the structured event data by applying multi-head attention. In this setup, the BERT embedding of the text span serves as the query, and the keys and values are the contextualized embeddings of the names and the one-hot encodings of the timestamps in the table. The resulting weighted sum of the one-hot encodings is taken logarithm and then added to the logits of the mean predictions. The base BERT of both models is initialized with the BlueBERT-Base Peng et al. [2020].

To fit the input length shorter than 128, the maximum sequence length of BlueBERT, we filtered out paragraph-level events that were annotated in the second pass. The fourteen annotation files[2] are split into 5 groups to perform 5-fold cross-validation, and we report the average of test performances across three different random seeds. For all the experiments, we trained the models for 20 epochs with a learning rate of 5e-5. For comparison, we also report the performance of the majority selection baseline, the baseline using Llama-2 (llama-2-13b-chat-hf) on the annotated clinical notes (same as Unimodal BERT) to generate the LB and UB through few shot prompting Touvron et al. [2023], and the baseline selecting the LB and UB based on the ground truth selection of the structured events.

4.3 Results

4.3.1 Comparison across different modalities

An exploratory data analysis of the 10 paired annotation files (5 multimodal and 5 unimodal) revealed a total of 2718 annotations, of which 693 were in Bounds-mode and 2025 were in Probability-mode. For each record, the number of annotations and their distribution between Bounds and Probability modes were very similar between the unimodal and multimodal versions. Furthermore, 95.7% of the events were annotated as exact matches across versions, meaning that the selected text had the same start and end positions, *i.e.*, span, regardless of annotation mode (Tab. 9a).

Across discharge summary sections, there were relatively more Bounds-mode annotations in the Medical History (67% bounds) and Discharge (35%) sections, and fewer in the larger Hospital Stay (7%) and Examinations and Findings (17%), indicating increased ability to annotate intraencounter events in Probability-mode. Across sections, there was high agreement between unimodal and multimodal annotation, shown by the proportion of exact matches (Tab. 9b).

All matching events annotated in Probability-mode were used to perform a Wilcoxon signed-rank test. We compared the standard deviations annotated for lower bound, upper bound, and duration. The standard deviations reported in the multimodal version were significantly smaller than those reported in the unimodal version. This trend was consistent across the lower bound (p-value < 0.001), upper bound (p-value < 0.001), and duration (p-value < 0.001). In general, we increased the precision on average by a factor of 1.42, 1.36, and 1.13 for the lower bound, upper bound, and duration, respectively. These values suggest that the uncertainty in multimodal annotations was reduced by 42%, 36%, and 13%, compared to unimodal annotations for the corresponding types of bounds and duration (Fig. 12).

Similarly, we conducted a comparison of the standard deviations pertaining to bounds and duration types across various sections present in clinical notes (Fig. 12). Our findings indicate that, in the case of bounds,

[2]One additional annotation file that 2012 i2b2 dataset does not include is added.

Comparison of Annotation Practices Across Versions					
			Bounds-Mode	**Probability-Mode**	**Exact Matches**
Record[1]	**Version**	**Annotations**	# (%)	# (%)	# (%)
A	Unimodal	161	45 (28.0%)	116 (72.0%)	156 (96.9%)
A	Multimodal	165	46 (27.8%)	119 (72.1%)	156 (94.5%)
B	Unimodal	379	53 (14.0%)	326 (86.0%)	354 (93.4%)
B	Multimodal	380	60 (15.8%)	320 (84.2%)	354 (93.2%)
C	Unimodal	182	58 (31.9%)	124 (68.1%)	167 (91.8%)
C	Multimodal	180	58 (32.2%)	122 (67.8%)	167 (92.8%)
D	Unimodal	352	125 (35.5%)	227 (64.5%)	347 (98.6%)
D	Multimodal	361	115 (31.9%)	246 (68.1%)	347 (96.1%)
E	Unimodal	278	64 (23.0%)	214 (77.0%)	276 (99.3%)
E	Multimodal	280	69 (24.6%)	211 (75.4%)	276 (98.6%)
Total	**-**	**2718**	**693 (25.5%)**	**2025 (74.5%)**	**2600 (95.7%)**

[1]Record numbers have been redacted

(a) Across Versions

Comparison of Annotation Practices Across Sections					
			Bounds-Mode	**Probability-Mode**	**Exact Matches**
Section	**Version**	**Annotations**	# (%)	# (%)	# (%)
Medical History	Unimodal	244	161 (66.0%)	83 (34.0%)	234 (95.9%)
Medical History	Multimodal	249	167 (67.1%)	82 (32.9%)	234 (94.0%)
Exams and Findings	Unimodal	484	29 (6.0%)	455 (94.0%)	467 (96.5%)
Exams and Findings	Multimodal	489	32 (6.5%)	457 (93.5%)	467 (95.5%)
Hospital Stay	Unimodal	366	59 (16.1%)	307 (83.9%)	354 (96.7%)
Hospital Stay	Multimodal	369	62 (16.8%)	307 (83.2%)	354 (95.9%)
Discharge	Unimodal	248	96 (38.7%)	152 (61.3%)	234 (94.4%)
Discharge	Multimodal	248	87 (35.1%)	161 (64.9%)	234 (94.4%)

(b) Across Sections

Table 9: Comparison of annotation practices across unimodal and multimodal versions (A) and across discharge summary sections (B), with number of annotations per record/per section, their breakdown into Bounds-mode and Probability-mode, and the exact match between the versions.

annotations exhibited an improvement in precision in all the sections. However, for the duration, there was no observed change in uncertainty, except for the Medical History, Exams and Findings, and Discharge sections.

In the sensitivity analysis, the standard deviation effect estimates (lower, upper, and duration) remained close to sample estimate, *i.e.*, the effect of increased precision in the multimodal annotations was maintained. The estimated precision factors were for lower bound, upper bound, and duration: 1.42 (95% CI [1.34-1.51]), 1.36 (95% CI [1.28-1.44]), and 1.13 (95% CI [1.10-1.17]) respectively. The Wilcoxon sign-rank test again showed differences between both versions of annotation (p-value < 0.001).

4.3.2 Comparison with i2b2 dataset (2012)

The total number of events in the thirteen annotation files derived from the multimodal data is 3532, with 2721 in Probability-mode annotations. The corresponding i2b2 training data files have 1024 events ("EVENT"

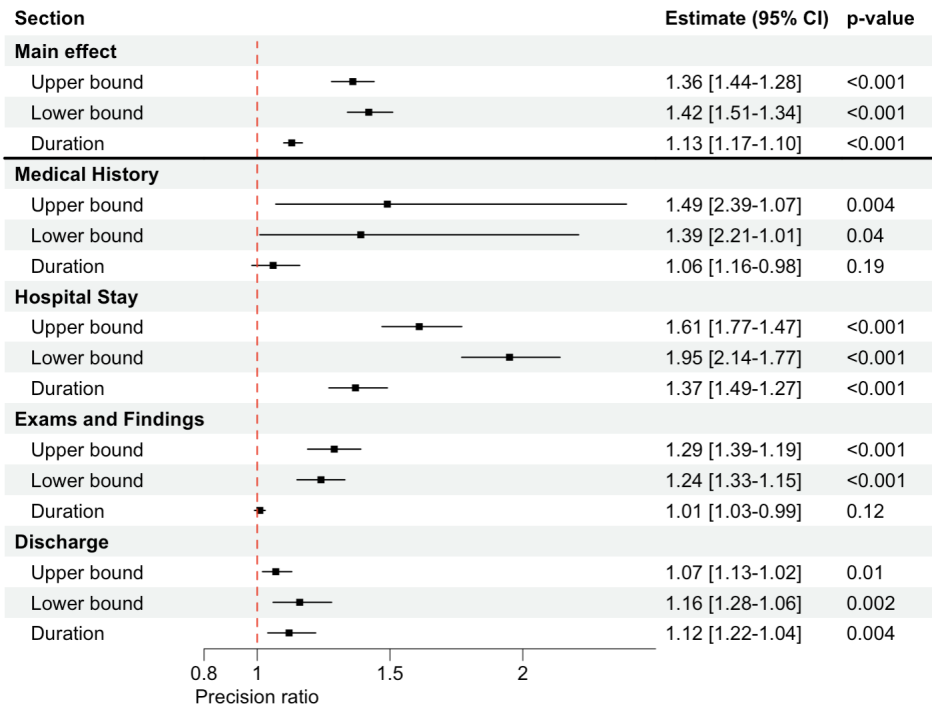


Figure 12: Precision factor between the two types of annotations. A value greater than 1 indicates a reduction of uncertainty in multimodal annotations (p-values obtained from the Wilcoxon sign-rank test).

tag) and 2140 temporal relations ("TLINK" tag), with 901 comparing events only. Note these i2b2 files do not cover the entire discharge summary. By aligning the text spans with the character-level IOU threshold of 0.5, 761 of the Probability-mode events are matched with the i2b2 events, and 490 of the i2b2 temporal relations can be mapped to our dataset. We use -1.0 for the threshold of z-scores to determine the temporal relations between two Probability-mode events. The confusion matrix between the i2b2 temporal relations and our annotations' temporal relations is shown in Fig. 4. The accuracy, the macro F1-score, and the Cohen's kappa score between the i2b2 and our temporal relations are 0.698, 0.599, and 0.383, respectively. In other words, 70% of the temporal links between the two events agree between our annotations and the i2b2 dataset. The reason for the low kappa score is imbalanced distribution of the categories, leading to a high expected agreement of random assignments.

To more thoroughly compare the temporal relations between the i2b2 dataset and ours, we conducted human assessment. For each type of mismatched relation, we randomly selected three examples, resulting in a total of 18 temporal relations. Then, a human with medical experience, who was not involved in the annotation process, evaluated the selected examples by examining the input note. This individual was asked to judge which annotation represented temporal relations more accurately. The results of the assessment are shown in Table 2. Out of the 18 examples, 9 were found to be better annotated in our dataset, while 8 were more accurately represented in the i2b2 dataset. This result suggests that the temporal relations solely from textual information are not complete, emphasizing the importance of incorporating structured data, but also suggests potential improvement in our multimodal based annotation process.

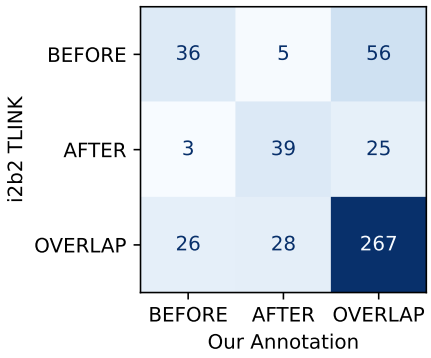


Figure 13: Confusion matrix between the temporal relations of the 2012 i2b2 dataset and our dataset

Temporal relation (i2b2 / Ours)	Judgment		
	i2b2	Ours	Both
BEFORE / AFTER	2	1	0
BEFORE / OVERLAP	2	1	0
AFTER / BEFORE	1	1	1
AFTER / OVERLAP	0	3	0
OVERLAP / BEFORE	2	1	0
OVERLAP / AFTER	1	2	0
Total	**8**	**9**	**1**

Figure 14: Human assessment on the 18 sampled mismatched temporal relations

Classification Version of Absolute Timeline Prediction								
	Type			Mean		Std Dev		
Method	**LB inf**[†1]	**UB inf**[†1]	**Anno**[†1]	**LB**[2]	**UB**[2]	**LB**[2]	**UB**[2]	**Dur**[2]
Tabular (Oracle)	-	-	-	0.594	0.789	-	-	-
Majority	0.859	1.000	0.840	0.224	0.238	0.220	0.207	0.267
Llama-2	-	-	-	0.374	0.441	-	-	-
Unimodal BERT	0.935	1.000	0.908	0.551	0.632	**0.446**	**0.447**	0.573
Multimodal BERT	**0.936**	1.000	**0.912**	**0.604**	**0.680**	0.433	0.436	**0.579**

[†]LB/UB inf: definiteness of LB/UB, Anno: Bounds- or Probability-mode
[1]Accuracy, [2]Macro-averaged F1 score

Table 10: Results of the classification version of absolute timeline prediction on our dataset. The best results among the non-oracle methods are highlighted in bold.

4.3.3 Multimodal Learning

Table 10 shows the results of the classification version of absolute timeline prediction. The F1 score of the lower bound prediction (Mean-LB) and the upper bound prediction (Mean-UB) of Multimodal BERT improved 10% (0.604 vs. 0.551) and 8% (0.680 vs. 0.632) from Unimodal BERT, respectively. This improvements, stemming from the integration of multi-head attention into the final mean prediction logits, demonstrate the benefits of integrating unstructured text with structured patient data. Comparing with Llama-2, our Multimodal BERT shows 61% (0.604 vs. 0.374) and 54% (0.680 vs. 0.441) higher F1 score in bound predictions. While the structured data baseline (Tabular (Oracle)) showed the best results in the upper bound mean prediction, this was under the idealized assumption of perfect attention on the structured data. The inferior lower bound mean prediction of the structured data baseline, compared to its upper bound, stems from the mismatch in labels between events anchored to the admission and the admission time itself.

4.4 Discussion and Conclusions

Clinical events from unstructured data provide information about the patient's progression, but placing them on an absolute timeline can be challenging. Yet while the timing of events in structured data is more certain, the structured data may miss events or other predictive insights. When both types of data are in alignment following the patient's clinical course, their combination may generate more complete and precise event timelines. Our work demonstrates the benefit of the multimodal approach, both in quality of annotation and prediction.

When comparing Probability-mode annotations, statistical analysis revealed superior precision for the multimodal version of the timeline in all three temporal entities (lower bound, upper bound, and duration). The precision factor for the bounds was similar at 1.40 and 1.34 (lower and upper, respectively), whereas for the duration it was at 1.13. This follows the intuition that the duration of many clinical events can be estimated based on clinical knowledge. In particular, the specific position of events on a timeline is more uncertain and depends on many factors that cannot be predicted with clinical knowledge alone, and having relevant structured data with precise timestamps greatly reduced timing uncertainty for the lower and upper bounds of events.

A subgroup analysis compared the precision factor of the timelines within different sections of the discharge summary. In the case of the bounds, multimodal annotation increased precision in all sections, but most prominently in the Hospital Stay and Medical History. The improvement in the Hospital Stay section is expected since the vast majority of the structured data parallels the patient's clinical course from admission to discharge. The Medical History section usually describes events that occurred prior to admission. Likewise, some of the structured data is generated during the patient's time in the emergency department, prior to admission. The Exam and Findings section also showed a moderate improvement in precision with the multimodal version. It contains laboratory tests and imaging studies, which could frequently be referenced to structured data with precise timestamps. Thus, the uncertainty of event timing is significantly reduced when the structured data is aligned with the unstructured data,

In the case of event duration, the multimodal version yielded statistically significant improvement in precision only in the Discharge and Hospital Stay section. Events in the Discharge section are very likely to have their upper bound anchored to the time of discharge, a event that was also available in the unimodal annotation. Thus, little improvement is seen in the precision of the upper bound when the rest of the structured data is made available. This aligns with the result that the upper bound experiences less improvement than the lower bound in the Hospital Stay since events here also extend until discharge.

This study has several limitations. First, one annotator was used, which precludes measurement of inter-annotator agreement. Additionally, the uncertainties defined for the bounds and duration could violate the positive semi-definite condition of a multivariate normal distribution, which may be oversimplified for the annotator's belief about the interval. Since a small sample size was used, the selected discharge summaries may not be representative, *e.g.*, due to differences in chief complaints, institutional policies, or note templates. Finally, there was no functionality for adding additional meta-data to events or recording temporal relations without established timelines.

The reported findings support the use of multimodal data to generate more precise event timelines when compared to unstructured data alone. The benefit is especially prominent when a large quantity of structured data aligns with the unstructured data. Further areas of study could include working with a larger sample size and analyzing differences when subjects are measured more frequently, *e.g.*, in critical care units versus the hospital floor.

The compatibility analysis with the i2b2 dataset validated that our dataset provides a complement to the existing text-based annotations. In the multimodal learning experiments, a BERT model leveraging structured events through multi-head attention improved F1 scores for predicting lower and upper bounds over the

unimodal BERT. This demonstrates how temporal localization of clinical events benefits from jointly modeling text and structured data sources. Overall, these strongly support the enhanced utility of our multimodal annotation approach for generating more precise absolute timelines of inpatient events.

Part II

Knowledge-Enriched Tabular Learning

Tabular problems are those where typical deep learning methods still struggle, and gradient-boosted trees continue to dominate [Grinsztajn et al., 2022], unlike other modalities like images, speech, and text. Because tabular data drive decisions in areas ranging from banking and logistics to clinical care and public policy forecasting, closing this gap would have enormous practical impact. We argue that a key reason for the gap is that neural models, despite their capability to generalize, treat a table as an anonymous matrix of numbers, ignoring the different types, scales, units and semantics of each column. In practice, every column carries deterministic metadata: a textual description, a unit, a type, or a known relationship to other columns. Current deep learning models for tabular data rarely leverage rich source of domain knowledge. Instead, they relearn obvious structure from scratch, often leading to over-fitting and poor calibration. This motivates our call for knowledge-enriched machine learning for tabular data.

In Section 5, we introduce *knowledge-enriched machine learning*, a framework that turns domain knowledge into algorithm-usable signals for tabular data. As a concrete instance we define kernel-enriched supervised learning, where a *concept kernel*, a kernel over columns encoding semantic relations among columns, supplements the standard training set. To make the idea reproducible, we release KE-TALENT, a benchmark of eleven diverse datasets, each bundled with column descriptions, embedding–based concept kernels, and training pipelines. Finally, we study two routes for turning a concept kernel into row-level geometry: i) smoothing the original inputs with respect to the kernel, and ii) constructing explicit value kernels over inputs that are explicitly specified in terms of the concept kernel. Our experiments show that kernel-enriched models demonstrate competitive performance and offer complementary feature representations, highlighting the potential and challenges.

The second strand of this part, developed in Section 6, asks a complementary question: instead of using concept kernels as an external source of structure, can we incorporate column semantics directly into a neural tabular model and exploit them to perform cross-table pre-training? To explore this, we design a *concept-conditioned* Transformer that represents each cell as a function of both its value and a learned embedding of the column's description, and we train it across the extended KE-TALENT benchmark with a combination of masked value prediction and contrastive objectives before fine-tuning on each dataset. In this "tabular foundation model" setting, pre-training improves downstream performance over training the same architecture from scratch and yields models that outperform neural baselines on average.

Taken together, the two lines of work in this part, kernel-enriched learning and concept-conditioned pre-training, map out a broader agenda for knowledge-enriched tabular modeling. The empirical results are mixed rather than transformative, but they demonstrate that incorporating domain knowledge can yield tangible gains and narrow, if not close, the gap to tree-based methods.

5 Concept Kernels for Column Semantics

This section is based on Kim et al. [2025a].

5.1 Introduction

In areas such as vision and language, deep learning has achieved remarkable success, benefiting from a combination of massive datasets and domain-appropriate inductive biases encoded into model architectures [Krizhevsky et al., 2012, Hochreiter and Schmidhuber, 1997]. However, in areas such as tabular, relational, and scientific machine learning, deep models lag behind tree-based methods like XGBoost [Grinsztajn et al., 2022]. This is often blamed on smaller dataset sizes in such settings. But as we argue in this paper, another critical reason is that the typical deep learning methods for tabular data do not leverage rich sources of deterministic domain knowledge, such as column names. Note that doing so is particularly challenging given the heterogeneity across datasets.

Overall, in many settings, rich sources of domain knowledge are indeed available, but what is lacking is a general approach for encoding such domain knowledge into algorithmically-usable forms. In this paper, we propose a general framework, which we call *knowledge-enriched machine learning*, to bridge this gap. This framework, specifically geared toward tabular data, provides high-level scaffolding for a new class of machine learning problems; as a paradigmatic example, we describe the problem of *kernel-enriched supervised learning* which extends the standard machine learning paradigm by considering an additional *concept kernel* as input to supervised learning algorithms.

To facilitate research in knowledge-enriched learning, we introduce KE-TALENT, a benchmark that extends the subset of the TALENT benchmark [Ye et al., 2024a] by incorporating structured metadata. Our benchmark consists of eleven datasets spanning diverse tasks and domains, with each dataset including column description and concept kernels derived from sentence embeddings to encode semantic relationship between columns. The codebase also provides training pipelines, enabling researchers to systematically compare different knowledge-enriched learning approaches.

Given a concept kernel that provides a notion of geometry over the individual table columns or attributes, a critical question is then how to translate to a notion of geometry over entire inputs (or table rows). As one class of approaches, we provide an implicit notion of geometry over inputs by considering various forms of smoothers of the input that smooth it with respect to the concept kernel. As another class of approaches, we explicitly construct so-called value kernels over inputs that are explicitly specified in terms of the concept kernel. Given geometry over inputs, we can then extract features and train performant supervised learning models.

Our experiments show that kernel-enriched models demonstrate competitive performance and offer complementary feature representations. These findings highlight both the potential and challenges of integrating deterministic knowledge into tabular learning.

The remainder of this section is structured as follows. Section 5.2 introduces the knowledge-enriched learning framework, formalizing the use of concept kernels. Section 5.3 describes KE-TALENT benchmark, detailing dataset construction and concept kernel generation. Section 5.4 presents our geometric approaches for knowledge-enriched learning. Section 5.5 provides experimental results, highlighting key findings and limitations. Finally, Section 5.6 concludes with a discussion of future research directions.

5.2 Knowledge-Enriched Learning Framework

We begin by introducing our general framework of *knowledge-enriched machine learning*. Broadly speaking, this framework focuses on machine learning algorithms which take structured forms of *deterministic information* as input, in addition to the standard input of a dataset.

Consider a standard supervised learning problem with input space $\mathcal{X}$ and output space $\mathcal{Y}$. A *supervised learning algorithm* is a (possibly random) mapping $\mathcal{A}\colon \mathcal{D} \mapsto \widehat{m}_{\mathcal{D}}$, where $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$ is a dataset and $\widehat{m}_{\mathcal{D}}\colon \mathcal{X} \to \mathcal{Y}$ is a prediction function. Put simply, a supervised learning algorithm $\mathcal{A}$ takes a dataset as input and returns a prediction function as output.

We can easily extend this definition to the general notion of a *knowledge-enriched supervised learning algorithm*, which again is a (possibly random) mapping $\mathcal{A}\colon (\mathcal{D}, \mathcal{I}) \mapsto \widehat{m}_{\mathcal{D}}$, where $\mathcal{I}$ is some structured form of deterministic information (domain knowledge). To specialize this framework, we must specify the structural form of the deterministic information $\mathcal{I}$. In choosing the structure of $\mathcal{I}$, one must carefully balance the following goals:

- **Flexibility:** The structure should be flexible enough to encode many different forms of commonly-available knowledge, such as logical rules, natural language descriptions, and relations from knowledge graphs.
- **Informativity:** The structure should be complex enough to carry problem-specific information.
- **Pragmatics:** The structure of $\mathcal{I}$ provides a pragmatic layer of abstraction between potentially highly-unstructured domain knowledge on one hand, and the practical constraints of algorithm design on the other.

In this paper, we focus on a simple but flexible encoding of deterministic information in the form *concept kernels*, designed with tabular data problems in mind.

5.2.1 Concept Domains and Values

Let $\mathcal{C}$ be an arbitrary set of concepts, e.g. in a table taken from a dating website's database, the concepts may correspond to column names, with $\mathcal{C} = \{\mathsf{age}, \mathsf{city}, \mathsf{headshot}, \mathsf{biography} \ldots\}$. Each concept $c \in \mathcal{C}$ is associated with a *concept domain*, denoted $\mathcal{V}_c$, which contains the possible values of that concept. In general, the concept domains may be both *rich* and *heterogenous*, e.g. we may have $\mathcal{V}_{\mathsf{headshot}} = [0,1]^{64\times 64\times 3}$ as the set of all 64×64 RGB images, and $\mathcal{V}_{\mathsf{biography}}$ as the set of all strings under 1,000 characters.

In these terms, a row in a table corresponds to assigning each concept to a value in its domain. When $\mathcal{C}$ and $(\mathcal{V}_c)_{c\in\mathcal{C}}$ are clear from context, we define $\mathcal{V} := \prod_{c\in\mathcal{C}} \mathcal{V}_c$ as the set of all possible assignments, termed the *value space*.[3] We use bold letters for assignments, and unbolded letters for values at specific concepts, e.g. $\mathbf{s} \in \mathcal{V}$ is an assignment and $s(c)$ is the value for concept c.

5.2.2 Concept Kernels

To relate concepts to a dataset $\mathcal{D}$, we require a correspondence between the concepts $\mathcal{C}$ and the input and output spaces $\mathcal{X}$ and $\mathcal{Y}$. Let $\mathcal{C}_{\text{in}} \subset \mathcal{C}$ be a set of *input concepts* and $\mathcal{C}_{\text{out}} \subset \mathcal{C}$ be a set of *output concepts*. Then, we assume that $\mathcal{X} = \prod_{c\in\mathcal{C}_{\text{in}}} \mathcal{V}_c$ and $\mathcal{Y} = \prod_{c\in\mathcal{C}_{\text{out}}} \mathcal{V}_c$, i.e., each $\mathbf{x} \in \mathcal{X}$ assigns each concept to a value in its domain, with $x(c) \in \mathcal{V}_c$ denoting the value assigned to concept c.

Finally, given a set of concepts $\mathcal{C}$, a *concept kernel* on $\mathcal{C}$ is a symmetric function $k\colon \mathcal{C} \times \mathcal{C} \to \mathbb{R}$. We are now ready to define a specific form of knowledge-enriched algorithm.

[3] In other contexts, such as physics and signal processing, the concept domains are homogeneous (i.e., $\mathcal{V}_c = \mathcal{V}_{c'}$ for all $c, c' \in \mathcal{C}$), in which case elements of $\mathcal{V}$ are called *configurations*, *states*, or *signals*.

Definition 1. *A* kernel-enriched supervised learning algorithm *is a (possibly random) mapping* $\mathcal{A}\colon \mathcal{D} \times k \mapsto \widehat{m}_{\mathcal{D}}$*, where* k *is a concept kernel over* $\mathcal{C}$*, and each point in* $\mathcal{D}$ *belongs to the value space* $\mathcal{V} = \prod_{c \in \mathcal{C}} \mathcal{V}_c$.

It is crucial to note that k is a kernel over *concepts* rather than a kernel over *values*, i.e. in the tabular setting, k measures the similarity between columns, not the similarity between rows. This fact distinguishes our setup from the traditional usage of kernels in machine learning, e.g. in Gaussian process regression.

Kernel-enriched Stochastic Processes In general, a kernel-enriched supervised learning algorithm is defined without any need to specify probabilistic assumptions on the dataset $\mathcal{D}$. However, for theoretical purposes, we must often specify a data-generating model for $\mathcal{D}$.

Hence, we define a *kernel-enriched stochastic process* on $\mathcal{C}$ as a pair $(k, \mathbf{S})$, where $k\colon \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ is a concept kernel, and $\mathbf{S} = (S(c))_{c \in \mathcal{C}}$ is a stochastic process, with the random variable $S(c)$ taking values in the domain $\mathcal{V}_c$. Then, letting $\mathbf{X} = (S(c))_{c \in \mathcal{C}_{\text{in}}}$ and $\mathbf{Y} = (S(c))_{c \in \mathcal{C}_{\text{out}}}$, we may assume that each pair $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ in $\mathcal{D}$ is an independent sample from $\mathbb{P}(\mathbf{X}, \mathbf{Y})$.

Constructing and Using Concept Kernels Thus far, we have discussed the general framework of knowledge-enriched machine learning, and a more specific instantiation: kernel-enriched supervised learning. However, several details remain. First, *how can concept kernels be constructed from existing sources of problem-specific information?* Second, *how can algorithms leverage concept kernels to improve performance?*

Both questions are quite open-ended and constitute entire possible areas of research. To seed these areas, the remainder of the paper offers several basic starting points. First, in Section 5.3, we introduce a benchmarking suite, which consists of several datasets along with potentially useful concept kernels. Then, in Section 5.4, we describe geometric approaches for leveraging concept kernels for the purpose of kernel-enriched machine learning.

5.3 KE-TALENT Benchmark

While a few tabular machine learning benchmarks exist [Grinsztajn et al., 2022], none explicitly incorporate deterministic information about column semantics. To address this, we introduce KE-TALENT, a benchmark that extends a subset of the TALENT benchmark [Ye et al., 2024a], the most recent large-scale collection of tabular datasets. Our benchmark enhances TALENT by including column descriptions and embeddings, facilitating the use of prior knowledge in ML models.

We selected eleven datasets from TALENT where descriptive column names or metadata are available from the original data sources. They cover a diverse range of tasks and configurations of numerical and categorical input features. The dataset selection was determined *prior* to running our method to prevent post-hoc selection bias. Table 11 provides an overview of the dataset statistics.

To facilitate the use of deterministic information, KE-TALENT includes the following for each dataset: (1) original and preprocessed dataset, (2) metadata for each concept (column), (3) raw sentence embeddings of each concept [Reimers and Gurevych, 2019], (4) several pre-computed concept kernels, and (5) code for preprocessing, training, and evaluation. The benchmark will be continuously expanded to further enhance its scope and applicability.

Concept kernels provided in KE-TALENT For a chosen dataset in our benchmark, let $\mathcal{C}$ denote its concepts, which are in one-to-one correspondence with column indices. As indicated, KE-TALENT includes concept embeddings $(\boldsymbol{\lambda}_c)_{c \in \mathcal{C}}$; these embeddings can be used to construct concept kernels $k(c, c')$ in several ways. We provide multipe types of concept kernels, including inner product, distance-based, and group-centered inner product kernels.

Dataset Name	Domain	Task	# class	# sample	# num	# cat
Abalone	Biology	reg	-	4177	7(7)	1(1/3)
Diamonds	Geology	reg	-	53940	6(6)	3(3/20)
Parkinsons Telemonitoring	Healthcare	reg	-	5875	18(18)	1(1/2)
Student Performance	Education	reg	-	651	1(11)	29(19/53)
Communities and Crime	Social Science	reg	-	1994	102(102)	0(0)
Bank Customer Churn	Business	bincls	2	10000	6(6)	4(4/9)
German Credit Data	Business	bincls	2	1000	7(7)	13(13/54)
Taiwanese Bankruptcy	Business	bincls	2	6819	95(95)	0(0)
ASP-POTASSCO	Computer Science	multicls	11	1294	140(140)	1(1/2)
Internet Usage	Social Science	multicls	46	10108	1(1)	69(69/423)
Student Dropout	Education	multicls	3	4424	17(17)	17(17/218)

Table 11: **Statistics of the datasets in KE-TALENT**. The **Task** column denotes the task type (*reg* = regression, *bincls* = binary classification, *multicls* = multi-class classification). The **# sample** column denotes the number of samples (rows). The **# num** column denotes the number of numerical columns, and the **# cat** denotes the number of categorical columns. Numbers in parentheses indicate the number of columns and categories after preprocessing.

5.4 Geometric Approaches & Empirical Results

A concept kernel specifies only a geometry over the *concepts* $\mathcal{C}$ (i.e., columns of a table), whereas in learning, we need to specify an inductive bias over the *values* $\mathcal{V}$ (i.e., rows of a table). Thus, a critical question in kernel-enriched learning is how to go from a geometry over coordinates to a geometry over values. Here, we describe several classes of approaches which accomplish this goal.

Notation For simplicity, we assume $\mathcal{C}$ is finite, with input concepts ordered as $\mathcal{C}_{\text{in}} = \{c_1, c_2, \ldots, c_D\}$ for some $D \in \mathbb{N}$. From here, we only use the concept kernel over the input concepts $k \colon \mathcal{C}_{\text{in}} \times \mathcal{C}_{\text{in}} \to \mathbb{R}$. Finally, we will frequently represent k by a symmetric matrix $\mathbf{K} \in \mathbb{R}^{D \times D}$, where $(\mathbf{K})_{ij} = k(c_i, c_j)$.

We assume *homogeneous* input concept domains, with values in some finite-dimensional vector space, i.e., $\mathcal{V}_c = \mathbb{R}^B$ for all $c \in \mathcal{C}_{\text{in}}$, which enables vector-space operations[4]. We can think of each dimension b as a *channel*. Then, we can identify each input value $\mathbf{x} \in \mathcal{X}$ with a matrix $\mathbf{M}_{\mathbf{x}} \in \mathbb{R}^{D \times B}$ whose i^{th} row equals $x(c_i)$. This makes $\mathcal{X}$ a vector space, with $\mathcal{X} \cong \mathbb{R}^{D \times B}$. We use $\mathbf{x}$ and $\mathbf{M}_{\mathbf{x}}$ interchangeably and write $\mathbf{x}_b$ for the b^{th} channel of $\mathbf{x}$ across all D concepts, i.e. $\mathbf{x}_b = (\mathbf{M}_{\mathbf{x}})_{:,b}$.

5.4.1 Smoothing Approaches

In the first class of approaches, we use the concept kernel to *implicitly* specify a geometry over values by specifying a transformation that takes each input value $\mathbf{x} \in \mathcal{X}$ to a smoother value $\widetilde{\mathbf{x}} \in \mathcal{X}$.

1. **Smoothness via kernel convolution** A simple way to smooth $\mathbf{x}$ with respect to k is via kernel

[4] $B = 1$ in typical tabular data, but we introduce B to generalize notation for cases where feature encodings are applied. If the input concept domains are heterogeneous, we can pre-process to make them homogeneous.

convolution in value space, i.e., via the transformation $(k * -)\colon \mathcal{X} \to \mathcal{X}$ defined as

$$(k * \mathbf{x})(c) := \sum\nolimits_{c' \in \mathcal{C}_{\text{in}}} k(c, c') \cdot x(c'), \tag{7}$$

Letting $\widetilde{\mathbf{x}} = (k * \mathbf{x})$, we can represent in matrices as $\mathbf{M}_{\widetilde{\mathbf{x}}} = \mathbf{K}\mathbf{M}_{\mathbf{x}}$. To preserve scale, we can apply either the row-normalized $\overline{\mathbf{K}}_{\text{row}} = \mathbf{D}^{-1}\mathbf{K}$, or the symmetric-normalized $\overline{\mathbf{K}} := \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}$ kernels, where $\mathbf{D}$ is the diagonal degree matrix $\mathbf{D}_{ii} = \sum_j \mathbf{K}_{ij}$. Notably, the value $\widetilde{\mathbf{x}} = \mathbf{D}^{-1}\mathbf{K}\mathbf{x}$ solves the following kernel-weighted least variance objective at each input concept c:

$$(\overline{k}_{\text{row}} * \mathbf{x})(c) = \operatorname*{arg\,min}_{v \in \mathcal{V}_c} \sum\nolimits_{c' \in \mathcal{C}_{\text{in}}} k(c, c') \cdot \|x(c') - v\|_2^2 \tag{8}$$

2. **Smoothness via regularization** Alternatively, one can use the concept kernel k to construct a smoothness penalty $R\colon \mathcal{X} \to \mathbb{R}$, and solve the regularized least squares problem

$$\widetilde{\mathbf{x}} = \operatorname*{arg\,min}_{\mathbf{v} \in \mathcal{X}} \sum\nolimits_{c \in \mathcal{C}_{\text{in}}} \|x(c) - v(c)\|_2^2 + \xi R(\mathbf{v}), \tag{9}$$

for a hyperparameter $\xi \geq 0$. For example, if k is a Mercer kernel, so that all eigenvalues in its spectral decomposition are nonnegative, then k induces the norm $R_{\text{norm}}(\mathbf{v}) := \|\mathbf{M}_{\mathbf{v}}^{\top}\mathbf{K}^{-1}\mathbf{M}_{\mathbf{v}}\|_2^2$. In this case, the optimum of Equation (9) is $\widetilde{\mathbf{x}}$ such that $\mathbf{M}_{\widetilde{\mathbf{x}}} = \mathbf{K}(\mathbf{K} + \xi I)^{-1}\mathbf{M}_{\mathbf{x}}$.

Alternatively, we can use the kernel-weighted distance a smoothness penalty, i.e., $R_{\text{lap}}(\mathbf{v}) := \sum_{c,c' \in \mathcal{C}} \overline{k}(c, c') \cdot \|v(c) - v(c')\|_2^2$, where $\overline{k}(c, c') = d(c)^{-1/2}k(c, c')d(c')^{-1/2}$. In this case, we perform and analyze in the spectral domain of $\overline{\mathbf{K}}$. The optimum of Equation (9) is $\widetilde{\mathbf{x}}$ such that $\mathbf{M}_{\widetilde{\mathbf{x}}} = ((1 + \xi)I - \xi\overline{\mathbf{K}})^{-1}\mathbf{M}_{\mathbf{x}}$.

5.4.2 Value Kernel Approaches

In the second class of approaches, we explicitly specify a geometry over the input space $\mathcal{X}$ by constructing a *value kernel* $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Given a concept kernel k, we start by using its spectral decomposition [5] to construct a corresponding feature map $\mathbf{\Phi}\colon \mathcal{C}_{\text{in}} \to \mathbb{R}^D$:

$$\mathbf{\Phi}(c) := \big(\sqrt{\lambda_1} \cdot \psi_1(c), \sqrt{\lambda_2} \cdot \psi_2(c), \ldots, \sqrt{\lambda_D} \cdot \psi_D(c)\big), \tag{10}$$

so that $k(c, c') = \langle \mathbf{\Phi}(c), \mathbf{\Phi}(c') \rangle$. This feature map projects each concept $c \in \mathcal{C}_{\text{in}}$ into a D-dimensional eigenspace associated with the concept kernel: $\boldsymbol{\phi}_m\colon c \mapsto \sqrt{\lambda_m}\psi_m(c)$ is the m^{th} component of $\mathbf{\Phi}$.

Given such a feature map $\boldsymbol{\phi}_m$, an input value $\mathbf{x} \in \mathcal{X}$ can be represented in terms of these concept features $\mathbf{x}_b = \sum_{m=1}^{D}(\alpha_{bm}/\sqrt{\lambda_m})\boldsymbol{\phi}_m$. This decomposition by $\boldsymbol{\phi}_m$ induces a value kernel

$$K(\mathbf{x}_b, \mathbf{x}'_b) = \sum\nolimits_{m=1}^{D} \frac{\alpha_{bm}}{\sqrt{\lambda_m}} \frac{\alpha'_{bm}}{\sqrt{\lambda_m}} = \sum\nolimits_{m=1}^{D} \frac{1}{\lambda_m} \langle \mathbf{x}_b, \boldsymbol{\psi}_m \rangle \langle \mathbf{x}'_b, \boldsymbol{\psi}_m \rangle = \langle \varphi(\mathbf{x}_b), \varphi(\mathbf{x}'_b) \rangle, \tag{11}$$

where φ is the *value feature map*, and we use the inner product $\langle \mathbf{f}, \mathbf{g} \rangle = \sum_{c \in \mathcal{C}_{\text{in}}} f(c) \cdot g(c)$ for any two functions $\mathbf{f}\colon \mathcal{C}_{\text{in}} \to \mathbb{R}$ and $\mathbf{g}\colon \mathcal{C}_{\text{in}} \to \mathbb{R}$.

[5] $k(c, c') = \sum_{m=1}^{D} \lambda_m \langle \psi_m(c), \psi_m(c') \rangle$ for orthonormal eigenfunctions $\boldsymbol{\psi}_m\colon \mathcal{C}_{\text{in}} \to \mathbb{R}$ and eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D$.

Different spectral transformations $s(\cdot)$ from the smoothing approach lead to different value kernels, but all share the same eigenfunctions ψ_m. Specifically, the choice of spectral transformation $s(\cdot)$ only introduces a point-wise scaling to each value feature component:

$$\varphi_s(\mathbf{x}_b) = \left(\frac{1}{\sqrt{s(\lambda_1)}} \langle \mathbf{x}_b, \psi_1 \rangle, \frac{1}{\sqrt{s(\lambda_2)}} \langle \mathbf{x}_b, \psi_2 \rangle, \cdots, \frac{1}{\sqrt{s(\lambda_D)}} \langle \mathbf{x}_b, \psi_D \rangle \right). \tag{12}$$

In practice, we set $s(\cdot) = 1$ when transforming input into value feature. This particular choice corresponds exactly to the Fourier coefficients with respect to the concept kernel.

Combining value kernels As described, difference choices of concept kernel k lead to different value kernels K. We can can combine entire value kernels K_1 and K_2 with *value* feature maps $\mathbf{\Phi}_1 \colon \mathcal{X} \to \mathbb{R}^{D \times B}$ and $\mathbf{\Phi}_2 \colon \mathcal{X} \to \mathbb{R}^{D \times B}$. We can concatenate the value feature maps, resulting in the value kernel $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}')$. Alternatively, we can convolve the value kernels as $K(\mathbf{x}, \mathbf{x}') = (K_1 * K_2)(\mathbf{x}, \mathbf{x}') := \int_{\mathcal{X}} K_1(\mathbf{x}, \mathbf{v}) \cdot K_2(\mathbf{x}', \mathbf{v}) \cdot d\mathbf{v}$, which amounts to $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ interacting through integration over an intermediate space.

5.4.3 Partially Specified Concept Kernels

Thus far, we have united several potential approaches to kernel-enriched supervised learning, showing that we can use the provided concept kernel k to construct a value kernel K. However, in many cases, the concept kernel may be only partially specified, e.g. a binary sparsity pattern for k as $\mathbf{B} \in \{0, 1\}^{D \times D}$. Also, even if the concept kernel is fully available, we might wish to better model the concept relationship by binarizing the kernel and learning from data.

In this scenario of a binarized kernel, we present this kernel as edges in a graph, where each concept acts as a node. Graph Neural Networks (GNNs) [Kipf and Welling, 2017], specifically Graph Attention Networks (GATs) [Veličković et al., 2017, Brody et al., 2022], naturally accommodate this setting. At each GAT layer, attention weights can be dynamically learned based on both current node features and concept embeddings. We refer to this variant as *Concept Graph Attention Networks* (CGATs). Formally, a CGAT layer with node features $\mathbf{H}^{(l)} \in \mathbb{R}^{D \times B}$ and concept embeddings $(\boldsymbol{\lambda}_c)_{c \in \mathcal{C}}$ updates node features as:

$$\mathbf{h}_c^{(l+1)} = \sum_{c' : B_{cc'} = 1} \alpha_{cc'} \mathbf{W}_t^{(l)} \mathbf{h}_{c'}^{(l)}, \text{ with } \begin{array}{l} \alpha_{cc'} = \exp(\text{score}_{cc'}^{(l)}) / \sum_{x : B_{cx} = 1} \exp(\text{score}_{cx}^{(l)}) \quad \text{and} \\ \text{score}_{cc'}^{(l)} = \mathbf{a}^{(l)\top} \sigma(\mathbf{W}_s^{(l)} \mathbf{h}_c^{(l)} + \mathbf{W}_t^{(l)} \mathbf{h}_{c'}^{(l)}) + (\mathbf{W}_e^{(l)} \boldsymbol{\lambda}_c)^\top (\mathbf{W}_e^{(l)} \boldsymbol{\lambda}_{c'}), \end{array} \tag{13}$$

where σ is a nonlinear activation, such as LeakyReLU, and $\mathbf{W}_s^{(l)}$,$\mathbf{W}_t^{(l)}$,$\mathbf{W}_e^{(l)}$, and $\mathbf{a}^{(l)}$ are learnable parameters. This approach can adaptively emphasize significant concept pairs based on both learned representations and concept kernels. We can also apply a multi-head attention scheme to this.

5.4.4 Using Concept Kernels for Self-Supervised Learning

Concept kernels quantify similarities between the concepts or columns in tabular data. Leveraging this property, we construct a self-supervised learning (SSL) objective by defining a *value transition distribution*, which systematically swaps or replaces column values through a concept-based Markov chain. Specifically, we row-normalize the concept kernel matrix[6] to form the concept transition matrix $\mathbf{T} = \mathbf{D}^{-1}\mathbf{K}$, which defines

[6]We assume $\mathbf{K}$ is non-negative, or pre-process it by clamping negative values to zero.

a Markov chain over concepts with a stationary distribution $\boldsymbol{\pi}_{\mathbf{K}}$. Given this, we define the value transition distribution as:

$$\mathbb{Q}(\mathbf{x}' \mid \mathbf{x}) = \sum_{c,c' \in \mathcal{C}} \boldsymbol{\pi}_{\mathbf{K},c} \mathbf{T}_{cc'} Q_{cc'}(x'_c | x_c), \tag{14}$$

where $Q_{cc'}$ is the value swap distribution to better capture complex or negative correlations.

Using this transition process, we generate multiple augmented views of training samples. SSL objectives, such as InfoNCE [Oord et al., 2018] or spectral contrastive loss [HaoChen et al., 2021], can then leverage these views to learn feature representations that inherently respect the geometric structure encoded by the concept kernel.

5.5 Experiments

Here, we evaluate the approaches of kernel-enriched learning discussed in Section 5.4 on KE-TALENT, and compare them with strong tabular ML baselines. First, we describe the implemented models of knowledge-enriched supervised learning and the baselines. Then, we outline the evaluation methodology to ensure a fair comparison. Finally, we present and analyze the results.

5.5.1 Models

Smoothing models These models construct smoothed representations of input values by applying transformations derived from the concept kernel. Specifically, we explore three smoothing methods: (1) convolution using the row-normalized kernel, (2) smoothing via norm regularization, and (3) smoothing with a Laplacian penalty. After smoothing, the resulting representations are provided as input to an MLP prediction model[7].

Value kernel model This method projects an input row onto spectral components derived from the concept kernels. We use two concept kernels: the inner product kernel and the group-centered inner product kernel, where concept groups are identified via HDBSCAN clustering. The model architecture consists of projection matrices where the input is transformed into the spectral features, concatenation with the original input value, and an MLP model.

Partially specified concept kernel (CGAT) In this approach, we construct a graph of columns where edges are the top-p[8] highest absolute values from the concept kernel matrix $\mathbf{K}$. This graph is then utilized by Concept Graph Attention Network (CGAT) described in Section 5.4.3. The model architecture includes: a feature encoding layer from RealMLP to transform each column value into a dense embedding, multiple GCAT convolution layers implemented with PyTorch Geometric [Fey and Lenssen, 2019], a max-pooling operation over node features, and an MLP prediction head.

Self-supervised learning model Our SSL model consists of a feature encoder, which transforms an input row into a fixed-length feature vector, and an MLP prediction head. We choose FT-Transformer [Gorishniy et al., 2021] as the feature encoder. Training the SSL model is in two steps: initial contrastive learning on the augmented dataset and finetuning on the supervised task.

[7] For the MLP architecture in smoothing and value kernel models, we re-implement RealMLP except data-driven initialization and dropout schedule.

[8] p is selected by hyper-parameter search.

Dataset	Abalone	Diamond	ParkTel	StuPerf	Crime	Churn	Credit	Taiwan	ASP	Internet	StuDrop
Method \ Task	reg/RMSE(↓)					bincls/Acc(↑)			multicls/Acc(↑)		
RealMLP	2.1210	523.92	**0.7337**	2.9277	0.1381	0.8735	0.7157	0.9667	0.3861	0.5302	0.7655
CatBoost	2.1789	524.91	1.5994	2.9244	**0.1336**	0.8759	**0.7430**	0.9718	0.3815	**0.5358**	**0.7782**
TabR	2.1078	**513.53**	8.0521	2.9072	0.1437	0.8743	0.7240	0.9678	0.3750	0.5183	0.7493
FT-T	2.1078	532.83	8.3437	2.9642	0.1369	0.8709	0.7123	0.9674	0.3678	0.5348	0.7547
Smooth(kernel)	2.1718	938.03	2.4700	3.0651	0.1466	0.8657	0.7160	**0.9722**	0.3815	0.5042	0.7162
Smooth(norm)	2.0879	903.70	1.2112	2.9725	0.1401	0.8688	0.6960	0.9659	**0.4013**	0.5093	0.7579
Smooth(Laplacian)	2.0937	522.37	0.9530	**2.8926**	0.1397	**0.8765**	0.7193	0.9694	0.3900	0.5259	0.7673
Value kernel	**2.0825**	525.79	0.8676	2.9203	0.1394	0.8746	0.7157	0.9673	0.3761	0.5315	0.7665
CGAT	2.0876	677.33	1.4612	3.0397	0.1425	0.8764	0.7337	0.9675	0.3838	0.5275	0.7656
SSL	2.1584	534.12	1.1275	2.9178	0.1373	0.8757	0.7180	0.9655	0.3964	0.5327	0.7571

Table 12: **Performance on KE-TALENT benchmark** The table reports test set performance of various baselines and kernel-enriched learning models. Results are averaged over 15 runs after hyper-parameter tuning. **Bold** indicates the best-performing method per dataset. Underlined values denote methods statistically indistinguishable from the best method using Welch's t-test with $p = 0.05$ (↓: lower is better, ↑: higher is better).

Dataset	CatBoost	SSL	CatBoost (+SSL Feature)
ASP-POTASSCO	0.3815	0.3964	**0.4067**

Table 13: **Combining SSL feature with CatBoost** Integrating SSL feature with CatBoost outperforms baseline CatBoost and SSL, suggesting that SSL representations complement tree-based models.

Baselines To assess the effectiveness of our methods, we compare the best models in their respective model classes on the TALENT benchmark: **RealMLP** [Holzmüller et al., 2024], a multilayer perceptron tailored for tabular data with tuned hyper-parameters; **CatBoost** [Dorogush et al., 2018], a gradient boosting method handling categorical features via target statistics; **TabR** [Gorishniy et al., 2024], which integrates a k-nearest-neighbor-like component into deep learning models; and **FT-Transformer** [Gorishniy et al., 2021], combining feature tokenizers with Transformer layers.

5.5.2 Evaluation method

Our evaluation methodology directly follows TALENT to ensure a fair and consistent comparison across models. For each dataset and knowledge-enriched model, we perform hyper-parameter optimization using Optuna Akiba et al. [2019] for 100 trials based on validation performance. Each model is trained 15 times with the optimal hyper-parameters to report average test performance. For the SSL model, hyper-parameter search is conducted only during the fine-tuning step, optimizing the MLP prediction head and training parameters, while the contrastive learning step follows the default FT-Transformer configuration from the TALENT codebase. For the baseline models, we use the results from the TALENT benchmark.

5.5.3 Results

Performance on KE-TALENT benchmark Table 12 presents performance of our kernel-enriched models and baselines on the KE-TALENT benchmark. CatBoost alone ranks best in four datasets, aligning prior

findings that gradient boosting excels in tabular data settings. Nonetheless, kernel-enriched methods also demonstrate competitive performance, showing the potential of knowledge-enriched learning: the smoothing models achieves top on four datasets, and the value kernel notably excels on Abalone dataset. Despite these successes, kernel-enriched models did not consistently surpass baselines, likely due to inefficiencies in concept kernel construction or how kernel information was integrated.

Analysis: Combining SSL features with CatBoost To assess the effectiveness of features learned from knowledge-enriched learning, we trained CatBoost using both the original tabular data and the feature from the pre-finetuning SSL encoder outputs on ASP-POTASSCO, where SSL ranked second. As shown in Table 13, this hybrid approach outperforms both baseline CatBoost and SSL, achieving the highest accuracy. This result suggests that SSL captures complementary metadata-driven features.

5.6 Discussion

Contributions In this paper, we outlined the general idea of *knowledge-enriched machine learning*, focusing on supervised learning algorithms that take an additional input in the form of a *concept kernel*. We proposed four meta-approaches that leverage concept kernels to inform their inductive biases. Additionally, we introduced KE-TALENT, a new benchmark for kernel-enriched supervised learning on tabular data, evaulated these approaches again strong tabular ML baselines. Our empirical results show that while kernel-enriched methods did not consistently outperform tree-based baseline, they demonstrated competitive performance and, in some cases, complementary feature representations.

Immediate future directions Our work opens the door to several future directions for knowledge-enriched machine learning in different domains. Here, we discuss a few directions of immediate importance, restricting our focus to the tabular data setting targeted by our benchmarking suite.

- **Improved kernel construction**: In our KE-TALENT benchmarking suite, we included several baseline kernels constructed from sentence embeddings. As the understanding of LLM embeddings improves, it may become possible to develop better methods for constructing kernels. Additionally, it may be interesting to use other forms of prior knowledge for constructing kernels, e.g. using knowledge graph embeddings [Ji et al., 2021].
- **Improved handling of heterogeneity**: Our datasets were carefully processed to ensure that the concept domains are relatively homogenous. In cases where this is not possible, or to further improve performance on these datasets, it may be necessary to enrich algorithms with other inputs, e.g. functions relating heterogeneous concept domains $\mathcal{V}_c$ and $\mathcal{V}_{c'}$.
- **Higher-order domain knowledge:** In some cases, binary relationships may not be sufficient to capture all domain knowledge. In these cases, it may be necessary to consider richer forms of knowledge enrichment and use those in methods which leverage higher-order structure, e.g. simplicial neural networks [Bodnar et al., 2021].

6 Concept-Conditioned Tabular Foundation Models

6.1 Introduction

Despite recent progress in deep learning for tabular data, strong tree-based ensembles such as CatBoost and XGBoost remain the dominant choice in practice, particularly on small- to medium-scale datasets with mixed numerical and categorical features. In contrast, language and vision have seen a rapid shift toward foundation models: large, self-supervised architectures that are pre-trained on massive corpora and then adapted to a wide range of downstream tasks. A natural question is whether a similar "tabular foundation model" can be built for heterogeneous tables, leveraging many datasets at once to learn general-purpose representations.

However, directly importing the foundation-model recipe into the tabular domain is challenging. Tabular datasets differ widely in source domain, schema, column data types, and value distributions, and current neural methods typically treat each table as an array of unnamed features. As a result, models trained on one dataset do not transfer well to others, and pre-training across multiple tables is non-trivial. Multi-dataset approaches such as XTab [Zhu et al., 2023] address this by introducing separate feature extractors or prediction heads per dataset, but this causes the number of parameters to scale with the number of tables and columns, limiting parameter sharing and generalization. Another line of work, in-context-learning-style tabular models, including TabPFN [Hollmann et al., 2025], trains Transformers to perform inference directly in-context on synthetic tasks generated from simple priors. While powerful, these methods rely on synthetic training distributions and do not explicitly exploit rich column semantics present in real-world tables.

At the same time, many tabular datasets come with human-readable metadata: column names, textual descriptions, and category labels that encode meaningful semantics (e.g., that `glucose` and `insulin` are related medical measurements, or that `churn` is the prediction target). Recent work such as CARTE [Kim et al., 2024] and TARTE [Kim et al., 2025b] begins to use such metadata, but typically relies on shallow word embeddings and does not fully decouple model parameters from specific tables. Moreover, these approaches do not directly incorporate detailed task descriptions or benefit from advances in modern instruction-tuned language models.

In this work, we propose a concept-conditioned tabular foundation model designed to address these limitations. Our model is a Transformer-encoder architecture that is explicitly parameterized by column semantics. It consists of three components: (i) a *concept-conditioned tokenizer* that converts each cell value and its column (or category) embedding into a token vector, (ii) a shared Transformer encoder that models interactions between columns via self-attention, and (iii) a *concept-conditioned predictor* that uses the same column embeddings to decode contextualized features into masked values or target outputs. Crucially, all three modules share parameters across all tables and datasets, and the number of trainable parameters does not depend on the total number of columns. Dataset- and column-specific information enters the model only through fixed embeddings extracted from a sentence-level language model.

We instantiate this idea using instruction-aware column and category embeddings obtained from Qwen3-Embedding-8B [Zhang et al., 2025], and train the model jointly across many heterogeneous tables. Pre-training combines masked value prediction with a contrastive row-level objective, followed by dataset-specific fine-tuning on each supervised target. To support this setting, we extend the KE-TALENT benchmark [Kim et al., 2025a] to 37 datasets with task, column, and category descriptions, substantially increasing both the number of concepts and the number of examples available for pre-training.

Empirically, we find that joint pre-training consistently improves downstream performance across model sizes, and that moderate-capacity models benefit the most. When compared against the 26 TALENT baselines, our best configuration (`small`, `pretrain`) ranks fourth overall—behind only CatBoost, LightGBM, and XGBoost—and achieves the best performance among all neural network models. This result is especially

significant considering that extended KE-TALENT heavily favors tree-based models. The remainder of this paper formalizes the multi-dataset problem setup, describes the model architecture and training procedure, introduces the extended KE-TALENT benchmark, and presents our experimental results.

6.2 Methods

In this section, we introduce our column-semantic, parameter-efficient tabular foundation model. We begin by formalizing the tabular prediction setting in the multi-dataset regime, assuming access to column-level semantic embeddings. We then describe the model architecture, which consists of a concept-conditioned tokenizer, a Transformer encoder, and a concept-conditioned value predictor. Finally, we present our pre-training and fine-tuning procedures across multiple heterogeneous tables.

6.2.1 Problem Setup

We build on the knowledge-enriched tabular learning framework of Kim et al. [2025a] and adopt similar notation. Let $\mathcal{C}$ denote a universe of *concepts*, which we identify with table columns, and let each concept $c \in \mathcal{C}$ have an associated domain $\mathcal{V}_c$ of possible values (e.g., real numbers or categorical labels). For a given table, the input concepts form a finite subset $\mathcal{C}^{\text{in}} \subset \mathcal{C}$, and there is an output (target) concept $c^{\text{out}} \in \mathcal{C}$. A row in the table is then an assignment $x \in \prod_{c \in \mathcal{C}^{\text{in}}} \mathcal{V}_c$ and a label $y \in \mathcal{V}_{c^{\text{out}}}$.

We consider a *multi-dataset* setting with M tabular datasets

$$\mathcal{D}_m = \big\{(x_m^{(i)}, y_m^{(i)})\big\}_{i=1}^{n_m}, \qquad m = 1, \ldots, M,$$

where each $\mathcal{D}_m$ has its own input and output concepts $\mathcal{C}_m^{\text{in}}$ and c_m^{out}. The configuration of each dataset, i.e., the number of columns $|\mathcal{C}_m^{\text{in}}|$, the mixture of numerical and categorical columns, and the number of examples n_m, may all vary across datasets.

We assume that for every concept $c \in \mathcal{C}$ we are given a fixed *column embedding* $\lambda_c \in \mathbb{R}^{d_e}$, obtained from textual descriptions of the column (e.g., column name, definition) using a sentence-level language model [Reimers and Gurevych, 2019, Zhang et al., 2025]. These embeddings encode semantic information about columns (e.g., that `Glucose` and `Insulin` are related), and can be used to construct a concept kernel $k(c, c') = \langle \lambda_c, \lambda_{c'} \rangle$ over columns. In contrast to prior work, which used k primarily as a fixed kernel to smooth or augment inputs [Kim et al., 2025a], we use the embeddings $\{\lambda_c\}$ directly inside a parametric encoder and prediction head. For a categorical column c, similarly to the column embeddings, we assume access to *category embeddings* $\{\eta_{c,k}\}_{k \in \mathcal{V}_c} \subset \mathbb{R}^{d_e}$ that embeds the semantics of each possible value.

Given a dataset $\mathcal{D}_m$, the supervised tabular prediction task is to learn a predictor $f_m : \prod_{c \in \mathcal{C}_m^{\text{in}}} \mathcal{V}_c \to \mathcal{V}_{c_m^{\text{out}}}$ that minimizes

$$\mathcal{L}_m^{\text{sup}}(f_m) = \mathbb{E}_{(x_m, y_m) \sim \mathcal{D}_m} \big[\, \ell\big(f_m(x_m), y_m\big) \,\big],$$

where ℓ is an appropriate task loss (e.g., L_2 for regression, cross-entropy for classification). Our goal is to learn a *single* tabular foundation model that can be pre-trained across $\mathcal{D}_1, \ldots, \mathcal{D}_M$ using only input columns, and then transferred and fine-tuned on each target dataset's supervised task.

6.2.2 Model Overview

Our model follows the Transformer-encoder style that has proven effective in language and vision foundation models [Devlin et al., 2018, Dosovitskiy, 2020] and recent tabular Transformers [Gorishniy et al., 2021, Zhu et al., 2023]. It consists of three modules:

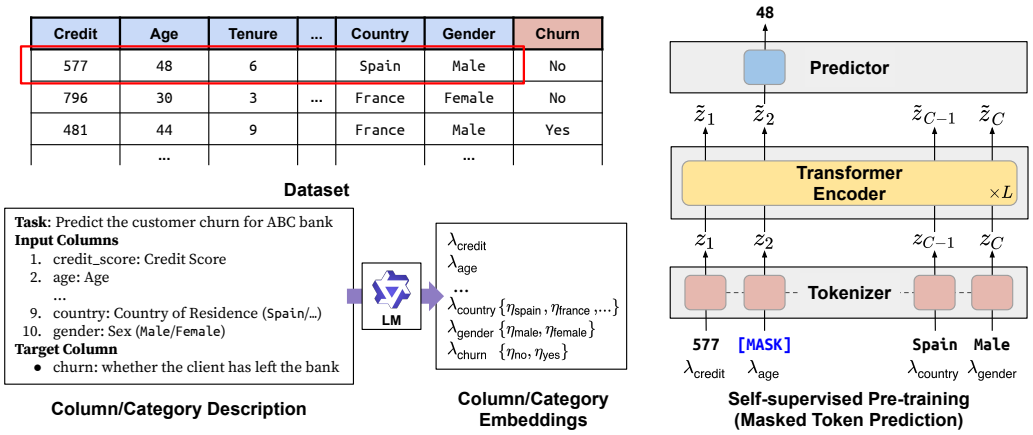


Figure 15: The overview of concept-conditioned tabular foundation model. Left: Given dataset with column name and description, we extract column and category embeddings from a pre-trained sentence-level language model. Right: Given an input row, we convert each column value and its column embedding into a feature vector (tokenizer), contextualize the per-column features (Transformer encoder), and predict the value of masked columns from their features (predictor).

1. **Concept-conditioned tokenizer** f_θ: For a given row x_m, the tokenizer maps each column value $v_c = x_{m,c}$ and its column embedding λ_c to a token vector $z_c \in \mathbb{R}^{d_{\text{token}}}$. The tokenizer operates *independently* per column and shares parameters across all columns, tables, and datasets.

2. **Transformer encoder** T_ϕ: The per-column token vectors $\{z_c\}_{c \in \mathcal{C}_m^{\text{in}}}$ are fed into a standard multi-layer Transformer encoder [Vaswani et al., 2017] that contextualizes each column representation via self-attention over all columns in the row. The encoder parameters are shared across datasets and do not depend on the number or identity of columns.

3. **Concept-conditioned predictor** g_ψ: Given the contextualized representations $\{\tilde{z}_c\}$ from the Transformer, the predictor uses both $\tilde{z}_c$ and λ_c to predict masked column values (for self-supervised pre-training) and target outputs (for supervised fine-tuning).

Crucially, the architecture is *parameter-efficient*: the number of trainable parameters in (θ, ϕ, ψ) does not scale with the total number of columns or datasets. There are no column-specific parameters, and all such column-specific information is injected only through the fixed column embeddings $\{\lambda_c\}$. Figure 15 shows the architecture of our concept-conditioned tabular foundation model.

6.2.3 Concept-Conditioned Tokenizer and Predictor

We now describe our concept-conditioned tokenizer f_θ and predictor g_ψ in more detail. Throughout, we write d_{token} for the token dimension.Figure 16 shows the architecture of the tokenizer and predictor.

Tokenizer Given a row x_m from dataset $\mathcal{D}_m$ and its input columns $\mathcal{C}_m^{\text{in}}$, the tokenizer processes each column $c \in \mathcal{C}_m^{\text{in}}$ independently, using its value $v_c = x_{m,c}$ and column embedding λ_c. We first map raw values v_c to a dense value embedding $h_c \in \mathbb{R}^{d_{\text{token}}}$ as follows:

- **Numerical columns**: For $v_c \in \mathbb{R}$, we apply a shared linear projection:

$$h_c = W_{\text{num}} v_c + b_{\text{num}}, \qquad W_{\text{num}} \in \mathbb{R}^{d_{\text{token}} \times 1},$$

  which yields a d_{token}-dimensional embedding that encodes the magnitude of the value but is agnostic to the column's semantics[9].

- **Categorical columns**: For v_c being one of a finite set of categories, we assume each category k comes with a category embedding $\eta_{c,k} \in \mathbb{R}^{d_e}$, constructed analogously to column embeddings. Given the active category k in the current row, we first map $\eta_{c,k}$ to token space via a two-layer MLP with a bottleneck:

$$h_c = W_2(W_1 \eta_{c,k} + b_1) + b_2,$$

  where $W_1 \in \mathbb{R}^{r \times d_e}$, $b_1 \in \mathbb{R}^r$, $W_2 \in \mathbb{R}^{d_{\text{token}} \times r}$, $b_2 \in \mathbb{R}^{d_{\text{token}}}$, $r \ll \min(d_e, d_{\text{token}})$. This low-rank projection is also used in concept-conditioned modulation which will be described below.

- **Masked columns**: For masked value prediction and target prediction, we mask some columns of the input. For these, we replace the value embeddings by a learnable mask embeddings $h_{\text{mask,num}}, h_{\text{mask,cat}} \in \mathbb{R}^{d_{\text{token}}}$.

To incorporate column semantics, we update h_c using the column embedding λ_c through a low-rank, feature-wise affine transformation reminiscent of FiLM [Perez et al., 2018] and parameter-efficient adapters [Hu et al., 2022]. Specifically, we compute scale and bias vectors $\gamma_c, \beta_c \in \mathbb{R}^{d_{\text{token}}}$ from λ_c via low-rank projections:

$$r_c = U\lambda_c + b_r \in \mathbb{R}^r, \quad \gamma_c = A_\gamma r_c + b_\gamma, \quad \beta_c = A_\beta r_c + b_\beta,$$

where $U \in \mathbb{R}^{r \times d_e}$, $b_r \in \mathbb{R}^r$, $A_\gamma, A_\beta \in \mathbb{R}^{d_{\text{token}} \times r}$, and $r \ll \min(d_e, d_{\text{token}})$ is a small rank. The concept-conditioned token embedding is then

$$z_c = \gamma_c \odot h_c + \beta_c,$$

with $\odot$ denoting element-wise multiplication. This information-bottleneck mechanism allows the model to explore low-dimensional linear projections from the space of language-model representations [Park et al., 2024b] in a parameter-efficient manner.

Transformer encoder Collecting the tokens for all input columns, we obtain a sequence $Z_m = \{z_c\}_{c \in \mathcal{C}_m^{\text{in}}} \in \mathbb{R}^{|\mathcal{C}_m^{\text{in}}| \times d_{\text{token}}}$, which we then feed to the Transformer encoder. We use a standard L-layer Transformer encoder [Vaswani et al., 2017] that acts over column tokens within a row:

$$\tilde{Z}_m = T_\phi(Z_m),$$

where $\tilde{Z}_m = \{\tilde{z}_c\}_{c \in \mathcal{C}_m^{\text{in}}}$ denotes the contextualized representations. The encoder models interactions between columns via multi-head self-attention.

Predictor The predictor g_ψ maps contextualized column features of the masked columns back to values. For each masked column c, we again apply concept-conditioned modulation:

$$\hat{z}_c = \gamma'_c \odot \tilde{z}_c + \beta'_c,$$

where (γ'_c, β'_c) are computed from λ_c via a low-rank projection in the same form as in the tokenizer. This serves as a light, concept-aware adapter before prediction.

[9] We pre-process each numerical column by standardizing using column mean and variance. Therefore, current implementation of h_c is a representation of the z-score

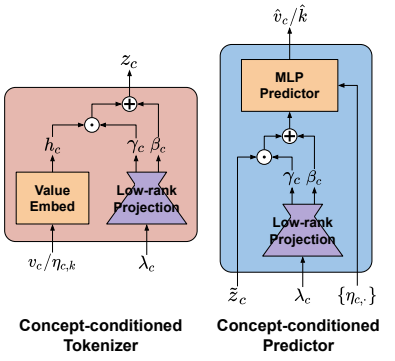


Figure 16: Concept-conditioned tokenizer and predictor. For both modules, features are infused with column semantics by column embeddings via feature-wise affine transformation.

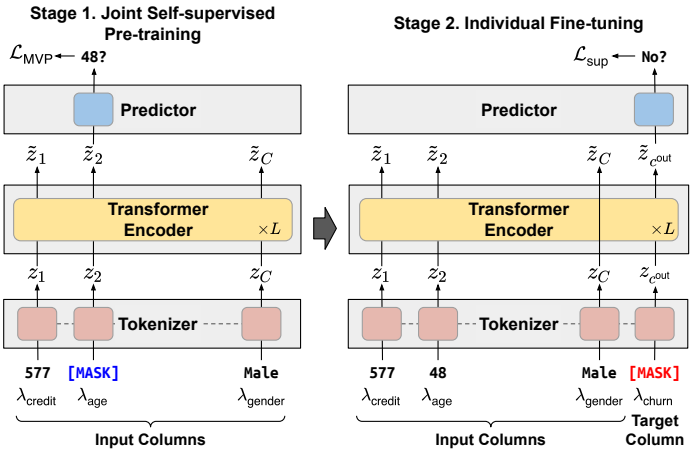


Figure 17: Training procedure of concept-conditioned tabular foundation model.

- **Numerical columns**: For a numerical column c, we predict a scalar via a shared linear head[10]:

$$\hat{v}_c = w_{\text{num}}^\top \hat{z}_c + b_{\text{num}}.$$

- **Categorical columns**: For a categorical column c, we cast prediction as multi-class classification. To remain parameter-efficient and consistent with the semantic embedding view, we obtain the weights for the final classification layer from the category embeddings $\{\eta_{c,k}\}_{k \in \mathcal{V}_c}$, similarly to the tokenization of categorical columns. The logits for column c are given by

$$\ell_{c,k} = \left\{W_2'(W_1'\eta_{c,k} + b_1') + b_2'\right\}^\top \hat{z}_c, \quad k \in \mathcal{V}_c,$$

We note that we use the same modules for target prediction. For a dataset $\mathcal{D}_m$, we treat the target column as an additional concept in $\mathcal{C}_m^{\text{in}} \cup c_m^{\text{out}}$, append it to the input row, mask its value, and apply the predictor to its contextualized representation. Thus the supervised predictor $\hat{y}_m = f_m(x_m)$ is simply a special case of masked value prediction applied to the designated target concept.

6.2.4 Training Procedure

We train the proposed model in two stages: (1) joint self-supervised pre-training and (2) fine-tuning on individual tables. Figure 17 shows the two-step training procedure.

Stage 1: Joint self-supervised pre-training In the first stage, we pre-train (θ, ϕ, ψ) using only input columns across all datasets $\{\mathcal{D}_m\}_{m=1}^M$ via a combination of masked value prediction and contrastive learning, in the spirit of BERT-style masked language modeling [Devlin et al., 2018]. Note that this self-supervised learning approach is already explored [Rubachev et al., 2022, Majmundar et al., 2022, Zhu et al., 2023], but none of them have provided a tokenization and prediction module shared across all tables.

[10] During training we predict the z-score of v_c with a squared loss objective. The evaluation metric (RMSE) is computed after rescaling to the original distribution.

For each row x_m, we sample a subset of columns $B(x_m) \subset \mathcal{C}_m^{\text{in}}$ to mask. The masked row $\tilde{x}_m$ replaces v_c by the mask token for $c \in B(x_m)$, while leaving other columns unchanged. We encode $\tilde{x}_m$ and predict the masked values $\{\hat{v}_c\}_{c \in B(x_m)}$. The masked value prediction loss is

$$\mathcal{L}_{\text{MVP}} = \sum_{m=1}^{M} \mathbb{E}_{x_m, B}\left[\sum_{c \in B(x_m)} \ell_{\text{rec}}\big(\hat{v}_c, v_c\big)\right],$$

where ℓ_{rec} is reconstruction loss (L_2 for numerical, cross-entropy for categorical).

In addition, we apply a row-level contrastive loss to encourage invariance under column-wise augmentations. For each row x_m, we generate two augmented views $\tilde{x}_m^{(1)}, \tilde{x}_m^{(2)}$ via noise injections.[11] We encode them to row representations $r_m^{(1)}, r_m^{(2)}$ via average pooling over contextualized column tokens, and minimize an InfoNCE loss [Oord et al., 2018]

$$\mathcal{L}_{\text{CL}} = -\sum_{m=1}^{M} \mathbb{E}_{x_m}\left[\log \frac{\exp(\langle r_m^{(1)}, r_m^{(2)} \rangle / \tau)}{\sum_{x'_m \in \text{batch}} \exp(\langle r_m^{(1)}, r_m'^{(2)} \rangle / \tau)}\right],$$

with temperature $\tau > 0$ and negatives drawn from other rows of the same dataset in the mini-batch. The total pre-training loss is $\mathcal{L}_{\text{pre}} = \lambda_{\text{MVP}} \mathcal{L}_{\text{MVP}} + \lambda_{\text{CL}} \mathcal{L}_{\text{CL}}$ with scalar weights $\lambda_{\text{MVP}}, \lambda_{\text{CL}}$.

Stage 2: Individual fine-tuning After self-supervised pre-training, we perform fine-tuning on individual datasets. In this stage, for a dataset $\mathcal{D}_m$, we introduce target information (i.e., target column embedding $\lambda_{c_m^{\text{out}}}$ and category embeddings $\{\eta_{c_m^{\text{out}},k}\}$ if the task is classification) to the model and train with a supervised loss

$$\mathcal{L}_{\text{sup},m} = \mathbb{E}_{(x_m, y_m)}\big[\ell(f_m(x_m), y_m)\big],$$

where f_m reuses the same (θ, ϕ, ψ) but put the mask token and applies the predictor to the target column c_m^{out}.

6.2.5 Models & Implementation Details

We evaluate four configurations of our concept-conditioned tabular foundation model, which we refer to as `tiny`, `small`, `base`, and `large`. These variants differ in the token dimension d_{token}, the number of Transformer encoder layers L, and the rank r of the low-rank concept-conditioned modulations. Table 14 summarizes the exact hyper-parameters used for each configuration.

We implement the concept-conditioned tokenizer and predictor as described in Section 6.2.3. Our implementation of Transformer encoder is based on the FT-Transformer backbone [Gorishniy et al., 2021].

To jointly training on multiple tables, we sample mini-batches from the 37 datasets using a mixture distribution over datasets. Let N_m denote the number of training examples in dataset $\mathcal{D}_m$. We define mixture weights $w_m \propto N_m^{\alpha}$ with exponent $\alpha = 0.5$. This choice interpolates between uniform sampling ($\alpha = 0$) and sampling proportional to dataset size ($\alpha = 1$), exposing smaller datasets more often. Within a batch, rows are sampled i.i.d. from the mixture over $\{\mathcal{D}_m\}$.

During joint self-supervised pre-training, we mask each column independently with probability 0.15 and use a perturbation probability of 0.20 to generate contrastive views. When perturbing numerical columns, we add Gaussian noise with standard deviation 0.2. For categorical columns, we randomly randomly replace

[11] We randomly select columns to perturb, inject Gaussian noise to numerical columns, and change categories of categorical columns randomly.

Table 14: **Configurations of the concept-conditioned models**. d_{token} is the hidden dimension, r is the low-rank dimension used in concept-conditioned modulation, L is the number of Transformer encoder layers, H is the number of attention heads, and d_{mask} is the hidden dimension of the predictor.

Model	d_{token}	r	L	H	d_{mask}	# Params
`tiny`	32	8	3	4	32	0.15M
`small`	64	16	4	4	128	0.44M
`base`	128	64	8	8	128	2.11M
`large`	256	256	16	8	256	12.83M

categories with other categories in the same column. We use the AdamW optimizer with a cosine learning rate schedule and linear warm-up (for self-supervised pre-training) or a constant learning rate (for fine-tuning).

During individual fine-tuning, we perform hyperparameter search using Optuna [Akiba et al., 2019]. Since the model architecture is fixed by the configuration, we only optimize over the learning rate and the weight decay for AdamW. For each dataset and model configuration, we run 100 trials of fine-tuning, selecting the best hyperparameters based on validation performance. All reported test metrics are the average of 15 runs with the best configuration.

6.3 Experiments

6.3.1 Extended KE-TALENT Benchmark

The hypothesis of knowledge-enriched tabular learning is that wider coverage of the *column embedding space* leads to better inductive biases for downstream tasks. In the original KE-TALENT benchmark, Kim et al. [2025a] curated 11 tabular datasets with column-level semantic metadata. However, the number of distinct concepts and domains was still limited, and many plausible relations between columns were not observed during training. In this work, we extend KE-TALENT to a larger and more diverse collection of tables in order to better probe and train concept-conditioned tabular models.

Starting from the 11 KE-TALENT datasets, we collect additional public tabular datasets from TALENT Ye et al. [2024a] benchmark that satisfy the following criteria: (1) they have meaningful input and target column names and, when possible, human-readable descriptions; (2) the number of input columns is not too small (at least 5); and (3) the task is non-trivial (e.g., k-NN or a linear model does not achieve near-perfect performance). This yields a total of 37 datasets. The number of distinct input concepts (columns) increases by a factor of 1.79 (from 538 to 962) and the number of examples increases by a factor of 4.58 (from 100k to 458k) in our extended benchmark. Table 15 summarizes the basic statistics of the extended benchmark, including the number of samples, numerical and categorical columns, and task type for each dataset.

Concept Embeddings For the column and category embedding extraction, we use `Qwen3-Embedding-8B` [Zhang et al., 2025] which is a instruction-aware embedding model. For each column $c \in \mathcal{C}_m^{\text{in}} \cup \{c_m^{\text{out}}\}$, we construct an input prompt that with a short task description (e.g., "Predict whether an employee will leave the company.") to obtain the column embedding $\lambda_c \in \mathbb{R}^{d_e} (d_e = 3072)$. Similarly, for each categorical value k in column c, we construct a phrase of the form "`column-name is category-name`" and encode it to obtain a category embedding $\eta_{c,k} \in \mathbb{R}^{d_e}$.

Dataset Name	Domain	Task	# class	# sample	# num	# cat
Abalone	Science	reg	-	4177	7	1(3)
Diamonds	Retail	reg	-	53940	6	3(20)
Parkinsons Telemonitoring	Healthcare	reg	-	5875	18	1(2)
Student Performance	Education	reg	-	651	1	29(53)
Communities and Crime	Social Science	reg	-	1994	102	0(0)
Bank Customer Churn Dataset	Business	bincls	2	10000	6	4(9)
German Credit Data	Business	bincls	2	1000	7	13(54)
Taiwanese Bankruptcy	Business	bincls	2	6819	95	0(0)
ASP-POTASSCO	Computer Science	multicls	11	1294	140	1(2)
Internet Usage	Social Science	multicls	46	10108	1	69(423)
Student Dropout	Education	multicls	3	4424	17	17(218)
1000-Cameras Dataset	Retail	reg	-	1038	10	0(0)
Basketball	Sports	bincls	2	1340	11	0(0)
Used Fiat 500	Retail	reg	-	1538	6	0(0)
Bias Correction	Science	reg	-	7725	21	0(0)
Brazilian Houses	Real Estate	reg	-	10692	8	0(0)
CPMP-2015	Computer Science	reg	-	2108	23	2(7)
CPS1988	Economics	reg	-	28155	2	4(10)
Cardiovascular Disease	Healthcare	bincls	2	70000	5	6(14)
Customer Personality Analysis	Marketing	bincls	2	2240	16	8(25)
E-Commerce Shipping	Logistics	bincls	2	10999	6	4(13)
Employee	Business	bincls	2	4653	4	4(10)
Fitness Club	Marketing	bincls	2	1500	3	3(15)
Food Delivery Time	Logistics	reg	-	45593	6	2(8)
INN Hotels Group	Hospitality	bincls	2	36275	12	5(20)
Job Profitability	Business	reg	-	14480	22	6(14)
Kaggle Bike Sharing Demand	Marketing	reg	-	10886	3	6(43)
MAGIC Telescope	Science	bincls	2	19020	9	0(0)
Miami Housing 2016	Real Estate	reg	-	13932	15	1(2)
NHANES Age Prediction	Healthcare	reg	-	2277	4	3(7)
Online News Popularity	Social Science	reg	-	39644	45	14(28)
Pima Indians Diabetes	Healthcare	bincls	2	768	8	0(0)
Superconductivity	Science	reg	-	21197	81	0(0)
Water Quality and Potability	Food	bincls	2	3276	8	0(0)
Wine Quality (Red)	Food	reg	-	1599	11	0(0)
Wine Quality (White)	Food	reg	-	4898	11	0(0)
Shop Customer Data	Marketing	reg	-	2000	4	2(12)

Table 15: **Statistics of the datasets in extended KE-TALENT**. The **Task** column denotes the task type (*reg* = regression, *bincls* = binary classification, *multicls* = multi-class classification). The **# sample** column denotes the number of samples (rows). The **# num** column denotes the number of numerical columns, and the **# cat** denotes the number of categorical columns. Numbers in parentheses indicate the number categories.

6.3.2 Experimental Settings

We now describe the experimental settings used to evaluate our concept-conditioned models and to compare them against the TALENT baselines.

Experiment 1: Ablations within our model family Our main experiment addresses two research questions: (i) how much does joint pre-training (Stage 1) improve downstream performance, and (ii) how does performance vary with model size?

(RQ1) Effect of joint pre-training We compare two training methods for each model configuration:

- `scratch`: (Stage 2 only) train the model on target prediction on each tabular dataset from scratch.
- `pretrained`: (Stage 1 $\rightarrow$ 2) initialize the model with the weights obtained from joint self-supervised pre-training, and fine-tune on the target prediction of each dataset.

This comparison isolates the contribution of joint pre-training to downstream performance.

(RQ2) Effect of model size. To study how capacity affects performance, we evaluate all four configurations in Table 14: `tiny`, `small`, `base`, and `large`, under both training regimes above. This allows us to assess whether increasing depth and width systematically improves performance and whether larger models benefit more from pre-training.

For Experiment 1, we follow Grinsztajn et al. [2022] and report the average of min–max normalized scores across datasets. Concretely, for each dataset we aggregate the performance of all 26 TALENT baselines with all of our model variants, map each model's score to $[0, 1]$ by min–max normalization, and then average the normalized scores over the datasets. All settings perform hyper-parameter search space for individual fine-tuning.

Experiment 2: Comparison with baselines In the second experiment, we compare our best-performing configuration against all 26 models from the TALENT benchmark, including strong tree-based, MLP-based, and Transformer-based methods such as CatBoost, LightGBM, XGBoost, RealMLP, TabR, and FT-Transformer. For each dataset, we rank all models (baselines and ours) by their test performance (rank 1 is best), then compute the average rank across the 37 datasets. This aggregate rank summarizes how often each method is competitive across diverse tabular tasks.

6.3.3 Results

Figure 18 and Table 16 summarize the ablations across model sizes and training regimes (Experiment 1). Figure 19 reports the average ranks comparing our best model against all TALENT baselines (Experiment 2).

Effect of joint pre-training Across all model capacities, joint pre-training provides consistent and meaningful improvements in downstream performance. For instance, (`tiny`,`scratch`) achieves a normalized score of 0.438, whereas (`tiny`,`pretrain`) improves to 0.473. The benefit becomes even more pronounced for larger models: (`base`,`scratch`) performs poorly (0.355), but (`base`,`pretrain`) attains one of the highest normalized scores (0.519). These results show that self-supervised joint training benefits downstream performance.

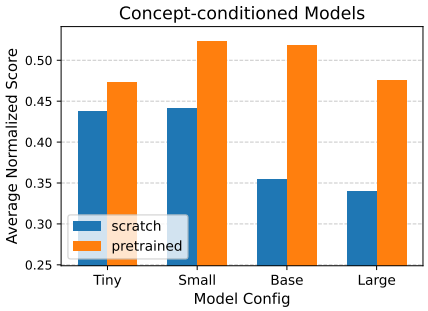


Figure 18: Effect of joint pre-training and model size.

Model	scratch	pretrain
tiny	0.4384	0.4733
small	0.4416	**0.5229**
base	0.3546	0.5185
large	0.3402	0.4757

Table 16: Normalized scores for our model variants under scratch versus pre-trained initialization. Pre-training yields substantial gains across all capacities.

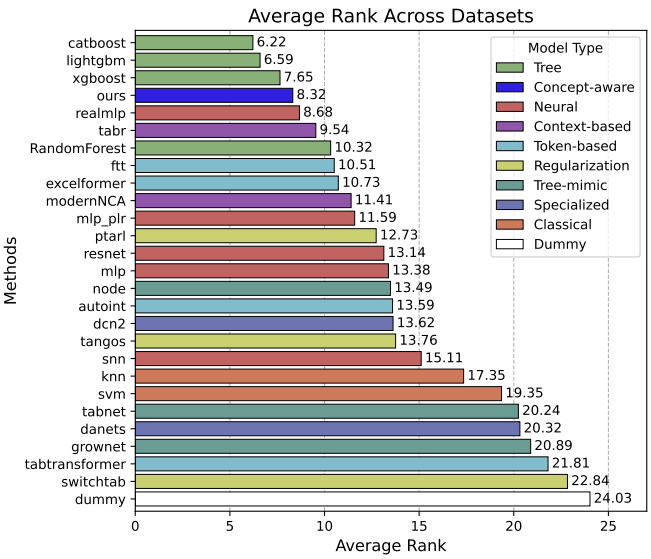


Figure 19: Average rank across the 37 KE-TALENT datasets comparing our best model (`small`, `pretrain`) against all TALENT baselines. Our model ranks fourth overall—behind only CatBoost, LightGBM, and XGBoost—and achieves the best performance among all deep-learning methods.

Effect of model size Model size exhibits a clear trend. Under the *scratch* setting, the `tiny` and `small` models outperform the larger `base` and `large` variants. With *pretraining*, performance improves from `tiny` to `small` and then declines, with the `large` model performing even worse than `tiny`. This pattern suggests that the larger models exceed the amount of learnable signal available in the extended KE-TALENT benchmark, leading to overfitting despite pretraining.

Comparison with baselines For Experiment 2, we compare our best-performing configuration, (`small`,`pretrain`), against all 26 TALENT baselines using the average rank across the 37 KE-TALENT datasets. A key characteristic of our benchmark is that it is more favorable to tree-based models: whereas the top methods in the full TALENT benchmark are RealMLP, MNCA, CatBoost, TabR, LightGBM, and XGBoost (in that order), the tree-based ensembles (CatBoost, LightGBM, and XGBoost) rank highest in our dataset selection.

Within this landscape, (`small`,`pretrain`) achieves an average rank of 8.32, placing it fourth overall and, critically, first among all neural network models. It outperforms heavily tuned MLPs (RealMLP), context-enhanced models (TabR), Transformer-based baselines (FT-Transformer, ExcelFormer), and all remaining neural architectures in the TALENT benchmark. This ranking is particularly meaningful given that RealMLP is the strongest baseline in the full TALENT benchmark.

Taken together, the results demonstrate that (i) joint pre-training is essential for achieving strong generalization across heterogeneous tabular tasks, and (ii) our approach achieves state-of-the-art performance among deep tabular models while remaining competitive with the strongest tree-based methods.

6.4 Conclusions

Contribution In this work, we introduced a "concept-conditioned tabular foundation model" that uses column and category semantics to enable parameter-efficient pre-training across many heterogeneous tables. The architecture consists of three shared components–a concept-conditioned tokenizer, a Transformer encoder, and a concept-conditioned predictor–that inject language-model embeddings of column and category descriptions into both the input and output pipelines. Because all parameters are shared across columns and datasets, the model size does not grow with the number of tables, making it naturally suited for joint training and transfer in the multi-dataset regime.

To support this setting, we extended the KE-TALENT benchmark to 37 datasets with task, column, and category descriptions, substantially increasing both the number of concepts and the total number of training examples. On this extended benchmark, we pre-trained our model with masked value prediction and contrastive objectives and then fine-tuned it on each dataset's supervised task. Empirically, joint pre-training consistently improved downstream performance across model sizes, and our best configuration (`small`, `pretrain`) achieved the top average rank among all neural baselines in TALENT while remaining competitive with strong tree-based ensembles such as CatBoost, LightGBM, and XGBoost.

Limitations This study has important limitations. Most notably, our approach currently applies only to tabular datasets with semantically meaningful column names and descriptions from which high-quality embeddings can be extracted. In many real-world settings, such metadata are missing, noisy, or inconsistent, which limits the immediate applicability of our method without additional curation or metadata generation.

Future Works These observations suggest several directions for future work. First, on the data side, we plan to further scale the benchmark by incorporating additional TALENT datasets with usable column descriptions, potentially expanding from 37 to on the order of 100 tables after careful screening. Second, on the modeling side, our framework can be combined with stronger numerical preprocessing techniques (e.g., power transformations or stretching of skewed features) to better match the inductive biases of tree-based methods on challenging regression problems. Third, there is room for architectural refinement of the tokenizer and predictor, such as tighter parameter sharing between the low-rank projections used for column embeddings and category embeddings, or alternative mechanisms for coupling language-model representations with value embeddings. We hope that these extensions (and the benchmark itself) serve as a foundation for future work on tabular foundation models that truly generalize across heterogeneous, semantically rich tables.

7 Conclusion

At a high level, this thesis has explored how *auxiliary information*–lexicons, syntactic structure, column descriptions, and other deterministic metadata–can be turned into useful inductive biases for neural models. Rather than treating these signals as peripheral, we asked when and how they can be integrated into otherwise standard architectures to improve accuracy, data efficiency, and interpretability in both NLP and tabular learning.

Learning with Auxiliary Information in NLP Across three studies, we showed that auxiliary information improves NLP systems in distinct ways.

- **Post-hoc spelling correction for clinical notes**: Combining a character-level language model with an explicit corruption model and a clinical vocabulary yielded a drop-in correction tool that reduced error rates compared to purely statistical baselines.
- **Guided compositional generalization**: Augmenting a Transformer with attention masks derived from parse trees provided just enough structural guidance to solve the hardest compositional splits that remain challenging for vanilla sequence models.
- **Multimodal event timelines**: Enriching discharge summaries with time-stamped EHR tables allowed a BERT-based encoder to localize event timing more precisely and to provide more interpretable temporal alignments.

Together, these results highlight three complementary roles for auxiliary signals: additional modalities, in-training guidance, post-hoc correction. Leveraging side signals (structured tables, syntactic masks, lexicons) proved to raise the task accuracy with little to no cost, expose interpretability handles that explain model behavior, and transfer to new domains where the same type of auxiliary resource exists.

Knowledge-Enriched Tabular Learning The second part of the thesis investigated tabular machine learning, where gradient-boosted trees and carefully tuned MLPs still dominate many benchmarks. We first introduced a knowledge-enriched framework that formalizes explicit column knowledge as *concept kernels*: kernels over columns that encode semantic relations between table attributes. On top of this framework, we released KE-TALENT, a benchmark of eleven public datasets packaged with column descriptions, concept embeddings, and ready-to-use training pipelines, and we analyzed how concept and value kernels induce useful row-level geometries. Building on KE-TALENT, we then proposed a *concept-conditioned tabular foundation model* that represents each cell as a function of both its value and a semantic embedding of its column, and extended the benchmark to 37 datasets to support cross-table pre-training. Using masked-value and contrastive objectives, we pre-trained this model across many heterogeneous tables and showed that, while it does not yet surpass the strongest tree-based baselines, pre-training consistently improves performance over training the same architecture from scratch and outperforms strong neural baselines on average. These results suggest that column semantics and cross-table structure can be translated into inductive biases that benefit neural tabular models, even if they are not yet sufficient to displace existing methods.

Limitations This work has several limitations. On the NLP side, the auxiliary resources we exploit–clinical lexicons, high-quality parses, aligned EHR tables–are not universally available and may be noisy or domain-specific, which limits the direct applicability of the methods to lower-resource settings. On the tabular side, both the concept-kernel and concept-conditioned approaches rely on sentence-level embeddings of column

and category descriptions which may not be available and their quality can vary widely and is difficult to evaluate directly. Moreover, despite empirical gains over comparable neural baselines, our tabular models still lag behind well-established tree-based and pre-tuned MLP methods on average performance. Finally, the computational budget for pre-training and hyperparameter search, while non-trivial, is modest compared to contemporary foundation models; more aggressive scaling and add more datasets may reveal behaviors that this thesis cannot fully characterize.

Future Work Several directions naturally follow from this thesis. In NLP, one avenue is to further integrate symbolic and structural information into large language models in a way that is compatible with in-context learning, for example by designing prompts or adapters that expose lexicons and parses without modifying the base model. In tabular learning, richer concept representations–such as combining textual descriptions with knowledge-graph structure or data-driven statistics–could yield more informative concept kernels and column embeddings. Scaling concept-conditioned pre-training to larger model families and broader table collections, and exploring hybrid architectures that combine knowledge-enriched encoders with gradient-boosted trees or other non-parametric learners, are promising steps toward closing the gap with current tabular state of the art. More broadly, the results in this thesis point to a design principle: auxiliary signals are rarely "free," but when they exist, carefully integrating them into model architecture and training can produce tangible benefits that purely data-driven approaches leave on the table.

References

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

J. Ainslie, S. Ontañón, C. Alberti, V. Cvicek, Z. Fisher, P. Pham, A. Ravula, S. Sanghai, Q. Wang, and L. Yang. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, 2020.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

J. Andreas. Good-enough compositional data augmentation. *arXiv preprint arXiv:1904.09545*, 2019.

G. Ang and E.-P. Lim. Guided attention multimodal multitask financial forecasting with inter-company relationships and global and local news. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, May 2022.

D. Bahri, H. Jiang, Y. Tay, and D. Metzler. SCARF: Self-supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations*, 2022.

A. Bardes, J. Ponce, and Y. Lecun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022.

C. Bodnar, F. Frasca, Y. Wang, N. Otter, G. F. Montufar, P. Lio, and M. Bronstein. Weisfeiler and Lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, 2021.

E. Brill and R. C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting of the association for computational linguistics*, pages 286–293, 2000.

S. Brody, U. Alon, and E. Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

T. T. Cheng, J. L. Cua, M. D. Tan, K. G. Yao, and R. E. Roxas. Information extraction from legal documents. In *2009 eighth international symposium on natural language processing*, pages 157–162. IEEE, 2009.

F. J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv.org*, Oct. 2018.

A. V. Dorogush, V. Ershov, and A. Gulin. CatBoost: Gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

A. Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

DWYL. List of english words. `https://github.com/dwyl/english-words`, 2020. Commit on Oct 15, 2020.

M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

P. Fivez, S. Šuster, and W. Daelemans. Unsupervised context-sensitive spelling correction of clinical free-text with word and character n-gram embeddings. In *BioNLP 2017*, pages 143–148, Aug. 2017. doi: 10.18653/v1/W17-2317. URL `https://www.aclweb.org/anthology/W17-2317`.

G. Frattallone-Llado, J. Kim, C. Cheng, D. Salazar, S. Edakalavan, and J. C. Weiss. Using multimodal data to improve precision of inpatient event timelines. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 322–334, 2024.

D. Furrer, M. van Zee, N. Scales, and N. Schärli. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *arXiv preprint arXiv:2007.08970*, 2020.

Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 2021.

Y. Gorishniy, I. Rubachev, N. Kartashev, D. Shlenskii, A. Kotelnikov, and A. Babenko. TabR: Tabular deep learning meets nearest neighbors. In *International Conference on Learning Representations*, 2024.

A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

L. Grinsztajn, E. Oyallon, and G. Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 2022.

J. Z. HaoChen, C. Wei, A. Gaidon, and T. Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems*, 2021.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S. B. Hoo, R. T. Schirrmeister, and F. Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

D. Holzmüller, L. Grinsztajn, and I. Steinwart. Better by default: Strong pre-tuned MLPs and boosted trees on tabular data. In *Neural Information Processing Systems*, 2024.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 2012.

E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

D. Hupkes, V. Dankers, M. Mul, and E. Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

S. M. Jayanthi, D. Pruthi, and G. Neubig. Neuspell: A neural spelling correction toolkit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 158–164, 2020.

S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

D. D. Johnson, A. E. Hanchi, and C. J. Maddison. Contrastive learning can find an optimal basis for approximately view-invariant functions. In *International Conference on Learning Representations*, 2023.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

M. D. Kemighan, K. Church, and W. A. Gale. A spelling correction program based on a noisy channel model. In *COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*, 1990.

D. Keysers, N. Schärli, N. Scales, H. Buisman, D. Furrer, S. Kashubin, N. Momchev, D. Sinopalnikov, L. Stafiniak, T. Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*, 2019.

J. Kim, P. Ravikumar, J. Ainslie, and S. Ontanon. Improving compositional generalization in classification tasks via structure annotations. 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)*, pages 637–645, 2021.

J. Kim, J. C. Weiss, and P. Ravikumar. Context-sensitive spelling correction of clinical text via conditional independence. In *Conference on Health, Inference, and Learning*, pages 234–247. PMLR, 2022.

J. Kim, C. Squires, and P. Ravikumar. Knowledge-enriched machine learning for tabular data. In *International Conference on Neuro-symbolic Systems*, 2025a.

M. J. Kim, L. Grinsztajn, and G. Varoquaux. CARTE: Pretraining and transfer for tabular learning. In *International Conference on Machine Learning*, 2024.

M. J. Kim, F. Lefebvre, G. Brison, A. Perez-Lebel, and G. Varoquaux. Table foundation models: on knowledge pre-training for tabular learning. *arXiv preprint arXiv:2505.14415*, 2025b.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012.

K. H. Lai, M. Topaz, F. R. Goss, and L. Zhou. Automated misspelling detection and correction in clinical free-text records. *Journal of biomedical informatics*, 55:188–195, 2015.

B. Lake and M. Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR, 2018.

A. Leeuwenberg and M.-F. Moens. Towards extracting absolute event timelines from english clinical reports. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2710–2719, 2020.

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, July 2020. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703.

H. Li, Y. Wang, X. Liu, Z. Sheng, and S. Wei. Spelling error correction using a nested rnn model and pseudo training data. *arXiv preprint arXiv:1811.00238*, 2018.

X. Li, H. Liu, and L. Huang. Context-aware stand-alone neural spelling correction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 407–414, 2020.

A. Liška, G. Kruszewski, and M. Baroni. Memorize or generalize? searching for a compositional rnn in a haystack. *arXiv preprint arXiv:1802.06467*, 2018.

S. Liu, X. Wang, and et al. Multimodal data matters: language model pre-training over structured and unstructured electronic health records. *IEEE Journal of Biomedical and Health Informatics*, 27(1):504–514, 2022.

C. J. Lu, A. R. Aronson, S. E. Shooshan, and D. Demner-Fushman. Spell checker for consumer language (cspell). *Journal of the American Medical Informatics Association*, 26(3):211–218, 2019.

K. Majmundar, S. Goyal, P. Netrapalli, and P. Jain. Met: Masked encoding for tabular data. *arXiv preprint arXiv:2206.08564*, 2022.

A. Moldwin, D. Demner-Fushman, and T. R. Goodwin. Empirical findings on the role of structured data, unstructured data, and their combination for automatic clinical phenotyping. *AMIA Summits on Translational Science Proceedings*, 2021.

N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, May 2023.

A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

K. Park, Y. J. Choe, Y. Jiang, and V. Veitch. The geometry of categorical and hierarchical concepts in large language models. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024a.

K. Park, Y. J. Choe, and V. Veitch. The linear representation hypothesis and the geometry of large language models. In *International Conference on Machine Learning*, 2024b.

Y. Peng, S. Yan, and Z. Lu. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets. In *Proceedings of the 2019 Workshop on Biomedical Natural Language Processing (BioNLP 2019)*, pages 58–65, 2019.

Y. Peng, Q. Chen, and Z. Lu. An empirical study of multi-task learning on bert for biomedical text mining. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 205–214, 2020.

E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

I. Rubachev, A. Alekberov, Y. Gorishniy, and A. Babenko. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208*, 2022.

P. Ruch, R. Baud, and A. Geissbühler. Using lexical disambiguation and named-entity recognition to improve spelling correction in the electronic patient record. *Artificial intelligence in medicine*, 29(1-2):169–184, 2003.

J. Russin, J. Jo, R. C. O'Reilly, and Y. Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*, 2019.

B. Schölkopf. Learning with kernels: support vector machines, regularization, optimization, and beyond, 2002.

T. M. Seinen, E. A. Fridgeirsson, and et al. Use of unstructured text in prognostic clinical prediction models: a systematic review. *Journal of the American Medical Informatics Association*, 29(7):1292–1302, 2022.

P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, 2018.

W. Sun, A. Rumshisky, and O. Uzuner. Annotating temporal information in clinical narratives. *Journal of biomedical informatics*, 46:S5–S12, 2013a.

W. Sun, A. Rumshisky, and O. Uzuner. Evaluating temporal relations in clinical text: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association*, 20(5):806–813, 2013b.

M. Tayefi and et al. Challenges and opportunities beyond structured data in analysis of electronic health records. *Computational Statistics*, 2021.

A. Templeton. *Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet*. Anthropic, 2024.

H. D. Tolentino, M. D. Matters, W. Walop, B. Law, W. Tong, F. Liu, P. Fontelo, K. Kohl, and D. C. Payne. A umls-based spell checker for natural language processing in vaccine safety. *BMC medical informatics and decision making*, 7(1):1–13, 2007.

H. Touvron, L. Martin, and et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint*, 2023.

Ö. Uzuner, B. R. South, S. Shen, and S. L. DuVall. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556, 2011.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Z. Wang, L. Gui, J. Negrea, and V. Veitch. Concept algebra for (score-based) text-controlled generative models. *Advances in Neural Information Processing Systems*, 2024.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. 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*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

H.-J. Ye, S.-Y. Liu, H.-R. Cai, Q.-L. Zhou, and D.-C. Zhan. A closer look at deep learning on tabular data. *arXiv preprint arXiv:2407.00956*, 2024a.

H.-J. Ye, H.-H. Yin, and D.-C. Zhan. Modern neighborhood components analysis: A deep tabular baseline two decades later. *arXiv preprint arXiv:2407.03257*, 2024b.

R. Zhai, B. Liu, A. Risteski, J. Z. Kolter, and P. K. Ravikumar. Understanding augmentation-based self-supervised representation learning via RKHS approximation and regression. In *International Conference on Learning Representations*, 2024.

Y. Zhang, M. Li, D. Long, X. Zhang, H. Lin, B. Yang, P. Xie, A. Yang, D. Liu, J. Lin, F. Huang, and J. Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Z. Zhong and D. Chen. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 50–61, 2021.

B. Zhu, X. Shi, N. Erickson, M. Li, G. Karypis, and M. Shoaran. Xtab: Cross-table pretraining for tabular transformers. *arXiv preprint arXiv:2305.06090*, 2023.

Appendix A A Symbolic–Statistical Approach for Clinical Spelling Correction

A.1 Data Processing

This section describes the data pre-processing procedure of the two real datasets in Section 2.5 and the reproduction of the spelling correction methods suggested in the dataset papers.

MIMIC-III Misspelling Dataset The MIMIC-III misspelling dataset is the only misspelling dataset based on the MIMIC-III clinical notes. However, there are several issues regarding their pre-processing code[12]. The code is based on an older version, v1.3, of the MIMIC-III database and splits processes text by the lines in the `NOTEEVENTS.csv` file, which leads to the risk to include the context from other notes in an example. After reviewed by a medical professional, we found that the correction labels of 30 examples are incorrect. We update these labels, some of which are multiple, and if an output is one of them, we marked it as correct. We have released the pre-processing code which is compatible with the latest version of the MIMIC-III database and revises the incorrect labels (link).

The baseline result in Table 2 is reproduced after running the word2vec training and hyper-parameter tuning. The numbers are not matched with Fivez et al. [2017] because of the revised labels and the randomness of the training and tuning.

CSpell Spelling Error Dataset The CSpell dataset contains various spelling errors: `Grammatical`, `Misspelling`, `Punctuation`, `RealWord`, `ToMerge`, and `ToSplit`. These spelling errors or their corrections can be multiple-word or contain non-alphabet characters or non-text entries, such as HTML entries. We only choose the examples in which the input word and the correction are single-word and alphabet-only.

As mentioned, CSpell software performs both detection and correction of spelling errors, and the input to the CSpell does not require the location of misspelled words. To find out which input word is detected, we use the output from the debug mode activated by the "`-d`" command-line flag. To make a fair comparison to our model, we excluded the CSpell test examples that the misspelled words are not detected by any of the detection modules of CSpell. Note that such filtering is not done on the CSpell training set to prevent our model from fitting to the CSpell's selection bias. As result, 409 and 574 examples are chosen from the training set and the test set of 1050 and 1924 examples, respectively.

The input texts of the CSpell dataset, consumer health questions, can contain multiple spelling errors, each of which constitutes an example. When we evaluate each misspelling, other misspellings in the same input text are corrected to clean the context and remove the interference of them. The output of CSpell software is evaluated only based on whether the typo words have been corrected properly, regardless of the other words.

A.2 Training, Tuning and Evaluation

This section describes the training procedure, the hyper-parameter search, and the final evaluation of our spelling correction method. Our character-level language model is based on the BART [Lewis et al., 2020] implementation of Hugging Face's Transformer [Wolf et al., 2020]. Both the Base and Large models are trained for 500k iterations on the MIMIC-III clinical notes with batch size 256. We trained the model on 4×NVIDIA A100 GPU 40GB. We follow the optimizer and the learning rate schedule same as the original

[12] `https://github.com/clips/clinspell`

Hyper-param	Values
Early stopping	$\{25000n : n = 1, ..., 20\}$ steps
C	$\{2.0, 5.0, 10.0, 20.0\}$
n	$\{0, 1, 2, 3, \infty\}$

Table 17: List of decoding hyper-parameters evaulated

Hyper-param	CSpell Training								Synthetic	
	LM only		LM + Dict		LM + ED		LM+ED+Dict			
	Base	Large	Base	Large	Base	Large	Base	Large	Base	Large
Early stopping	475k	450k	475k	450k	475k	475k	475k	300k	500k	500k
C	-	-	-	-	5.0	5.0	5.0	5.0	5.0	5.0
n	-	-	-	-	1	1	1	1	2	2

Table 18: The result of hyper-parameters search on different datasets and settings

BERT, except that the learning rate for the encoder part is reduced to one-tenth of the decoder since the encoder is finetuned from BlueBERT.

We tuned the hyper-parameters on the correction accuracy on the CSpell training set. Hyper-parameters of our method are only from the beam search decoding: the loss weight C and the number of characters ahead n in the corruption model. Also, for early stopping, we evaluated our method for every 25k steps of the training of the character-level language model. For the full model and each of the ablation studies, we did the grid search on all possible values of C, n, and training steps, summarized in Table 17. The selected hyper-parameter values are shown in the first eight columns of Table 18.

The time complexity of the beam search is proportional to the beam width B. $B = 30$ is chosen during the hyper-parameter tuning for faster search, and we use $B \in \{30, 300\}$ when evaluating our method on the CSpell test set and the Clinspell set for better correction outputs.

A.3 Subgroup Analysis by UMLS Semantic Type

This section describes the grouping procedure of subgroup analysis by UMLS Semantic Types. We first choose three subtrees from the hierarchy of the UMLS Semantic Types [13] with head nodes `substance`, `Pathologic Function`, and `Finding`. Each subtree becomes a subgroup of Semantic Types that represents a word category: "substance", "disease", and "symptom". The Semantic Types of each subgroup are as follows:

- Substance: `Substance`, `Pharmacologic Substance`, `Antibiotic`, `Biomedical or Dental Material`, `Biologically Active Substance`, `Hormone`, `Enzyme`, `Vitamin`, `Immunologic Factor`, `Receptor`, `Hazardous or Poisonous Substance`, `Organic Chemical`, `Amino Acid, Peptide, or Protein`, `Inorganic Chemical`, `Element, Ion, or Isotope`, `Body Substance`, `Food`
- Disease: `Pathologic Function`, `Disease or Syndrome`, `Mental or Behavioral Dysfunction`, `Neoplastic Process`

[13] `https://www.nlm.nih.gov/research/umls/META3_current_semantic_types.html`

- Symptom: `Finding`, `Laboratory or Test Result`, `Sign or Symptom`

Also, we made "other" subgroup for examples that do not belong any of three subgroups.

Then, we grouped examples into Subgroups with their correction words. Given an example, we retrieved CUIs (Concept Unique Identifiers) of its correction word and related Semantic Types by quering to UMLS API [14]. If any of the example's Semantic Types fall into a subgroup's Semantic Types, then we put the example into the subgroup. Note that examples can belong to more than one subgroup since a word can have multiple UMLS Semantic Types. For example, a word "depression" fall into both "disease" and "symptom" subgroups.

A.4 Results in Unsupervised Setting

This section describes the procedure to generate the MIMIC-III synthetic misspelling dataset and the hyper-parameter tuning results on it.

Due to the limited amount of public dataset of clinical typo, we build a dataset of synthetically generated misspellings. From the MIMIC-III clinical notes, we randomly choose words that are in our reference dictionary and corrupt them with random operations. To corrupt words, up to two operations of character addition, deletion, substitution, or transposition can be applied. As a result, we generated 10k examples of misspellings. In the 10k examples of syntactically generated data, 10% are unchanged from the original word, 45% are modified with a single operation (insertion, deletion, or substitution), and 45% are modified with two operations. As a result, out of 8970 error examples, 3199 and 5771 are real and non-word errors, respectively.

We performed the hyper-parameter search on this synthetic dataset, as we did with the CSpell training set. Since it takes much computation to evaluate CIM on the synthetic dataset, we did not search for early stopping and choose the final language model. The grid search was performed on the other hyper-parameters, C and n. The last two columns of Table 18 show the results.

A.5 Spelling Correction Example

Figure 20a to 20d show the beam search results of our spelling correction model on several examples of the CSpell test set. For each example, we display the top 10 beam candidates (out of 300) of our Base and Large model. Our model combines the language model score (`LM`) and the corruption model score (`ED`) and normalizes it to get the final beam score (`Score`).

For easy cases such as Figure 20a, both the language model and the corruption model gives a high score to the correct output. For the typo of a complex terminology like Figure 20b, there might be an incorrect word ("proctectomy") that has a smaller edit distance than the correct word ("prostatectomy"), but the language model gives a high score to the correct word.

Also, we report some failure cases of our method. Figure 20c shows a failure case where the language model fails. In Figure 20c, although both "having" and "facing" fit the context, the language model grants a much lower score to "facing", which makes "having" be the top candidate by the Base model. In Figure 20d, although the correct word "tightening" gets the higher LM score, the corruption model gives a lower score it. This implies that the corruption model needs to be improved to give a higher score to the correct word in such cases.

[14] We use the latest version of UMLS, which is 2021AA

Figure 20: Beam search decoding results for several examples of the CSpell test set. For each example, we display the top 10 beam candidates. The column next to the candidate (`Score`) shows the final beam score for each candidate.

[Input]
Typo / Correct : prpostectomy / prostatectomy
Context: "my husband had a simple retropubic **prpostectomy** 6 weeks ago. he can not walk wit. My husband had a simple r etropubic prostatectomy. he can not walk without the walker, he can not move his right leg alone"
[Output]

Base/475k/B300/ED1(1)	Score	LM	ED	Large/300k/B300/ED1(1)	Score	LM	ED
prostatectomy	-1.1696	-0.0982	-1.0714	**prostatectomy**	-1.1057	-0.0342	-1.0714
periostectomy	-1.5619	-0.8476	-0.7143	proctectomy	-1.4805	-0.6472	-0.8333
proctectomy	-1.8191	-0.9858	-0.8333	appendectomy	-2.2858	-0.3628	-1.9231
periosteotomy	-2.0782	-1.0068	-1.0714	proctostomy	-2.2902	-0.6235	-1.6667
parotidectomy	-2.1398	-0.3541	-1.7857	postectomy	-2.2912	-0.9275	-1.3636
appendectomy	-2.2063	-0.2832	-1.9231	proctocolectomy	-2.3218	-0.4468	-1.8750
postectomy	-2.2858	-0.9221	-1.3636	cystoprostatectomy	-2.3307	-0.2254	-2.1053
preostectomy	-2.4425	-2.0578	-0.3846	prostatotomy	-2.3428	-0.8043	-1.5385
parathyroidectomy	-2.4522	-0.2299	-2.2222	nephrectomy	-2.3527	-0.2694	-2.0833
pylorectomy	-2.4765	-0.8098	-1.6667	papillectomy	-2.3949	-0.8565	-1.5385

(a) A success example.

[Input]
Typo / Correct : syntoms / symptoms
Context: "bad huge blood clots during her menstrual cycles after she was prescribed Ocella for birth control. Also th ese **syntoms** worsened after she gave birth. This has been happening for a year now. should she see discuss this"
[Output]

Base/475k/B300/ED1(1)	Score	LM	ED	Large/300k/B300/ED1(1)	Score	LM	ED
symptoms	-1.2953	-0.1842	-1.1111	**symptoms**	-1.2335	-0.1223	-1.1111
systems	-2.5961	-1.3461	-1.2500	syndromes	-2.4693	-0.9693	-1.5000
syndromes	-2.6394	-1.1394	-1.5000	systems	-2.4899	-1.2399	-1.2500
syndrome	-2.8220	-1.1553	-1.6667	syndrome	-2.6580	-0.9913	-1.6667
symptom	-2.8439	-0.9689	-1.8750	sensations	-2.6691	-0.3964	-2.2727
sensations	-2.9676	-0.6948	-2.2727	bottoms	-2.7527	-0.8777	-1.8750
symptom's	-2.9678	-1.4678	-1.5000	symptom's	-2.7878	-1.2878	-1.5000
syncopes	-3.0207	-1.3540	-1.6667	symptom	-2.8031	-0.9281	-1.8750
symptomatics	-3.1481	-0.8404	-2.3077	symptomatics	-2.9592	-0.6516	-2.3077
syncope	-3.1973	-1.3223	-1.8750	sensors	-3.0031	-1.1281	-1.8750

(b) A success example.

[Input]
Typo / Correct : faceing / facing
Context: "A FEMALE AGED PATIENT OF 65 YEARS OLD IS suffering IN POLYUREA CONDITION SHE IS NOT A PATIENT OF diabetes S HE **faceing** A PROBLEM OF EXCESSIVE URINE AT NIGHT TO GET UP NO change IN THE COLOR CHANG BLOOD IN URINE BURNING FEVER"
[Output]

Base/475k/B300/ED1(2)	Score	LM	ED	Large/300k/B300/ED1(1)	Score	LM	ED
having	-3.2793	-1.1364	-2.1429	**facing**	-2.8995	-2.1852	-0.7143
facing	-3.3089	-2.5947	-0.7143	having	-3.0808	-0.9379	-2.1429
feeling	-3.3493	-1.4743	-1.8750	causing	-3.2121	-1.3371	-1.8750
hacking	-3.3700	-2.1200	-1.2500	receiving	-3.2722	-1.2722	-2.0000
causing	-3.3907	-1.5157	-1.8750	feeling	-3.3320	-1.4570	-1.8750
happening	-3.4183	-1.4183	-2.0000	baseline	-3.3624	-1.1402	-2.2222
falling	-3.4725	-2.2225	-1.2500	falling	-3.3726	-2.1226	-1.2500
finding	-3.5307	-1.6557	-1.8750	raising	-3.3795	-1.5045	-1.8750
passing	-3.5499	-1.6749	-1.8750	takeing	-3.4534	-2.2034	-1.2500
baseline	-3.5644	-1.3422	-2.2222	happening	-3.4825	-1.4825	-2.0000

(c) A failure example of the language model.

[Input]
Typo / Correct : tighting / tightening
Context: "abdominal aorta aneurysm. I have an aneurysm (2.6cm for the last 5yrs last checked about a year ago experie ncing pain, **tighting** in abdomen and chest should I be going to emergency room."
[Output]

Base/475k/B300/ED1(3)	Score	LM	ED	Large/300k/B300/ED1(2)	Score	LM	ED
lighting	-1.8270	-1.2715	-0.5556	lighting	-1.8186	-1.2631	-0.5556
righting	-1.9277	-1.3721	-0.5556	**tightening**	-1.8317	-0.9226	-0.9091
tightening	-1.9883	-1.0792	-0.9091	righting	-1.9845	-1.4290	-0.5556
nighting	-2.1890	-1.6335	-0.5556	fighting	-2.2743	-1.7188	-0.5556
tincting	-2.3586	-1.2475	-1.1111	nighting	-2.2746	-1.7191	-0.5556
sighting	-2.3814	-1.8258	-0.5556	lightening	-2.4226	-1.0589	-1.3636
bighting	-2.4258	-1.8702	-0.5556	tightens	-2.4664	-1.3553	-1.1111
tingling	-2.4553	-0.7886	-1.6667	sighting	-2.5331	-1.9775	-0.5556
dighting	-2.5457	-1.9901	-0.5556	tightened	-2.5589	-1.0589	-1.5000
tightness	-2.5537	-0.5537	-2.0000	rightening	-2.5678	-1.2042	-1.3636

(d) A failure example of the corruption model.

Appendix B Structured Attention for Compositional Generalization

B.1 Full Experimental Results

In this section, we report the full results on the CFQ classification dataset and the structure annotation experiments. In all configurations, multiple evaluation metrics (accuracy, F1 score, and AUC) are computed by averaging the results of two randomly initialized experiments. We test each network using only the val set, not the test set, since the main purpose of the experiment is to compare the compositional generalization ability, not to select best hyper-parameter. Accuracy and F1 score are computed with the threshold 0.5 of the softmax output of label 1.

All the experiments of the CFQ classification datasets were run using the TensorFlow [Abadi et al., 2016] framework. As we explain in the Section 3.5, we use the ETC Transformer Ainslie et al. [2020] code for relative position embeddings. For the Transformer implementation, we use the code provided in a Tensorflow tutorial. The training is run on the `n1-highmem-8` instance (52GB RAM, 8 virtual cpus) of Google Cloud Platform, extended with NVIDIA Tesla V100 GPUs.

Hyper-parameters used in the training of neural networks are listed in Table 19. One thing that we want to clarify is that training steps are required number of steps to converge and the training did not last longer than needed. Nevertheless, the experiments with structure annotations required more training steps than LSTM/Transformer, especially when the network is using hard mask. We conjecture that training with the hard mask of parse trees is slow since only a small part of the attention is not masked and hence propagating the gradient via supervision at the <CLS> position is slow.

B.1.1 The CFQ classification Dataset

Table 20 shows the classification results of various methods of generating classification datasets, including one additional configuration (*MCD Split & Random Negatives*). The dataset generated by this new configuration has the train and the dev/test set that have different compound distributions, because it is based on the MCD split. However, because of the method used in generating negative instances (random negatives), the binary classification of correspondence can be easily generalizable to the dev set.

B.1.2 Structure Annotation

One possible annotation of the input structure is a mask to allow tokens of the question and SPARQL queries to only attend within their segment. We call this mask as *block attention* and test it as an alternative to the hierarchical attention structures (parse trees). This mask is denser than the attention mask from parse trees and sparser than "no mask". Figure 21 shows the block attention for the examples shown in the Figure 7.

Table 21 reports the full results of experiments on structure annotations. In all cases, entity cross links improve compositional generalization on the dev set, but provide a significant gain only when combined with the parse tree attention and the attention is guided by the "hard mask". As we can see in the "hard mask" experiments, block attention does not improve compositional generalization, which suggests a need for more detailed attention mask of input structure.

B.2 Related Works on Compositional Generalization

In this section, we review prior works on improving compositional generalization in more detail.

Russin et al. [2019] proposed to split the attention mechanism into two separate parts, syntax and semantics. The semantic part encodes each token independent of the context (this is a pure embedding look-up table),

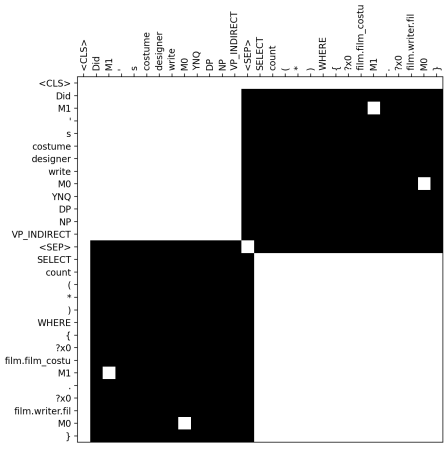


Figure 21: Block attention mask for the CFQ classification example of Figure 7. The dots at top-right and bottom-left are from entity cross links.

and the syntactic part encodes each token by looking only at its context (without looking at the token itself). In this way, the syntactic part tries to capture the syntactic role a token might play in a sequence. They show improved compositional generalization on the SCAN dataset using LSTMs, with respect to using standard attention. Compared to Russin et al. [2019] that uses LSTMs for the syntactic part, we use Transformer architecture to handle the hierarchical structure of the input.

In their follow up work on the CFQ dataset, Furrer et al. [2020] showed that an increased amount of pre-training helped Transformer models better generalize compositionally.

Another idea that has been proposed is to augment the training data, adding synthetic training examples to give the model a compositional learning bias [Andreas, 2019] .

Finally, work also exists on using general-purpose models like *Neural Turing Machines* or *Differential Neural Computers* Graves et al. [2016] that are often trained via reinforcement learning to solve compositional generalization tasks. These models learn an "algorithm" that can solve the task at hand, rather than trying to learn a direct input/output mapping as the Transformer models used in most other works do.

B.3 Examples of the CFQ classification dataset

In Figure 22, we present more examples of the CFQ classification datasets. In all cases, the random negative queries substantially differ from the positive queries, implying that a learner can easily perform the task. On the other hand, the model negative queries only differ by a token or a phrase, which demands a learner's higher ability.

	LSTM	Transformer	ETC
Hidden layers	2	{2,6}	6
Last dense layers	2	1	1
Hidden Size	512	128	128
Filter size	-	2048	512
Number of heads	-	16	16
Dropout	0.4	0.1	0.1
Batch size	1024	512	112
Training steps			
Random & Random	20k	10k	-
MCD & Random	20k	10k	-
MCD & Model	30k	20k	200k
Optimizer	Adam (0.85, 0.997)	Adam (0.9, 0.997)	Adam (0.9, 0.997)
Learning rate schedule	Constant	Inverse sqrt	Inverse sqrt
Base learning rate	0.001	0.001	0.001
Warmup steps	-	1000	1000
Weight decay	0.0	0.0	0.0

Table 19: Hyper-parameters used in training deep neural networks on the CFQ classification datasets

Dataset 1: *Random Split & Random Negatives*

Model	**Train**			**Train (hold-out)**			**Dev**		
	Acc	F1	AUC	Acc	F1	AUC	Acc	F1	AUC
LSTM	0.9999	0.9998	1.0000	0.9984	0.9967	0.9998	0.9982	0.9964	0.9998
Transformer (2 layers)	0.9988	0.9976	0.9998	0.9982	0.9964	0.9997	0.9988	0.9975	0.9998
Transformer (6 layers)	0.9992	0.9988	0.9999	0.9989	0.9978	0.9999	0.9990	0.9979	0.9999

Dataset 2: *MCD Split & Random Negatives*

Model	**Train**			**Train (hold-out)**			**Dev**		
	Acc	F1	AUC	Acc	F1	AUC	Acc	F1	AUC
LSTM	0.9999	0.9998	1.0000	0.9982	0.9965	0.9999	0.9546	0.9025	0.9923
Transformer (2 layers)	0.9982	0.9965	1.0000	0.9974	0.9948	0.9999	0.9942	0.9883	0.9996
Transformer (6 layers)	0.9986	0.9972	0.9999	0.9979	0.9958	0.9997	0.9889	0.9775	0.9991

Dataset 3: *MCD Split & Model Negatives*

Model	**Train**			**Train (hold-out)**			**Dev**		
	Acc	F1	AUC	Acc	F1	AUC	Acc	F1	AUC
LSTM	0.9990	0.9979	1.0000	0.9796	0.9604	0.9972	0.8226	0.5199	0.8310
Transformer (2 layers)	0.9817	0.9639	0.9988	0.9592	0.9202	0.9931	0.8359	0.5835	0.8789
Transformer (6 layers)	0.9886	0.9776	0.9995	0.9582	0.9189	0.9931	0.8414	0.6191	0.8738

Table 20: Results of the CFQ classification dataset generated with different CFQ splits and negative example strategies

Model	Mask Type	Parse Tree	Block Attn	Cross link	Train			Train (hold-out)			Dev		
					Acc	F1	AUC	Acc	F1	AUC	Acc	F1	AUC
LSTM	-				0.9990	0.9979	1.0000	0.9796	0.9604	0.9972	0.8226	0.5199	0.8310
Transformer	-				0.9886	0.9776	0.9995	0.9582	0.9189	0.9931	0.8414	0.6191	0.8738
Transformer w/ structure annotations (ETC)	No	-			0.9874	0.9751	0.9994	0.9591	0.9199	0.9934	0.8434	0.6202	0.8868
	Hard	Y	N	N	0.9955	0.9911	0.9999	0.9766	0.9543	0.9978	0.8628	0.6744	0.9061
		Y	N	Y	0.9978	0.9956	1.0000	0.9866	0.9738	0.9992	0.9170	0.8269	0.9656
		N	Y	N	0.9828	0.9659	0.9989	0.9567	0.9152	0.9928	0.8324	0.5874	0.8771
		N	Y	Y	0.9871	0.9746	0.9993	0.9573	0.9171	0.9930	0.8386	0.6048	0.8881
	Soft	Y	N	N	0.9863	0.9728	0.9993	0.9588	0.9197	0.9933	0.8426	0.6017	0.8729
		Y	N	Y	0.9891	0.9784	0.9995	0.9603	0.9226	0.9936	0.8482	0.6385	0.8819
	Hard +Soft	Y	N	N	0.9940	0.9882	0.9999	0.9743	0.9500	0.9973	0.8615	0.6697	0.9056
		Y	N	Y	0.9975	0.9949	1.0000	0.9867	0.9739	0.9991	0.9249	0.8473	**0.9721**

Table 21: Results of the CFQ classification dataset (MCD split & model negatives) with different types of structure annotations

• Question (CFQ input)

```
Did M0 ' s founder produce M1
```

• Positive query (CFQ output)

```
SELECT count ( * ) WHERE {
 ?x0 film.producer.film~ M1 .
 ?x0 ~organizations_founded M0
}
```

→ Label: 1 (same)

• Model negative query

```
SELECT count ( * ) WHERE {
 ?x0 film.producer.film~ M0 .
 ?x0 ~organizations_founded M1
}
```

→ Label: 0 (different)

• Random negative query

```
SELECT count ( * ) WHERE {
 ?x0 a film.film .
 ?x0 film.film.edited_by ?x1 .
 ?x0 film.film.edited_by M3 .
 ?x0 film.film.edited_by M4 .
 ?x0 film.film.written_by M1 .
 ?x0 film.film.written_by M2 .
 ?x1 a film.actor
}
```

→ Label: 0 (different)

(a)

• Question (CFQ input)

```
Did a British sibling of M0 direct M2
```

• Positive query (CFQ output)

```
SELECT count ( * ) WHERE {
 ?x0 film.director.film M2 .
 ?x0 ~.person.nationality m_07ssc .
 ?x0 people.person.sibling_s~ M0 .
 FILTER ( ?x0 != M0 )
}
```

→ Label: 1 (same)

• Model negative query

```
SELECT count ( * ) WHERE {
 ?x0 film.director.film M2 .
 ?x0 ~person.nationality m_07ssc .
 ?x0 people.person.sibling_s~ M2 .
 FILTER ( ?x0 != M0 )
}
```

→ Label: 0 (different)

• Random negative query

```
SELECT count ( * ) WHERE {
 ?x0 a film.editor .
 ?x0 film.cinematographer.film M3 .
 ?x0 ~.person.gender m_05zppz .
 ?x0 ~.person.nationality m_03_3d
}
```

→ Label: 0 (different)

(b)

• Question (CFQ input)

```
Which male person directed M2 , M3 , and M4
```

• Positive query (CFQ output)

```
SELECT DISTINCT ?x0 WHERE {
 ?x0 a people.person .
 ?x0 film.director.film M2 .
 ?x0 film.director.film M3 .
 ?x0 film.director.film M4 .
 ?x0 people.person.gender m_05zppz
}
```

→ Label: 1 (same)

• Model negative query

```
SELECT DISTINCT ?x0 WHERE {
 ?x0 a people.person .
 ?x0 film.director.film M2 .
 ?x0 film.director.film M3 .
 ?x0 film.director.film M4 .
 ?x0 people.person.gender m_02zsn
}
```

→ Label: 0 (different)

• Random negative query

```
SELECT count ( * ) WHERE {
 ?x0 a film.actor .
 ?x0 film.editor.film M1 .
 ?x0 film.editor.film M2 .
 ?x0 ~films_executive_produced M3 .
 ?x0 film.writer.film ?x1 .
 ?x1 a film.film
}
```

→ Label: 0 (different)

(c)

• Question (CFQ input)

```
Who directed a film , executive produced M1 and M2 , and executive produced M3 and M4
```

• Positive query (CFQ output)

```
SELECT DISTINCT ?x0 WHERE {
 ?x0 a people.person .
 ?x0 film.director.film ?x1 .
 ?x0 ~films_executive_produced M1 .
 ?x0 ~films_executive_produced M2 .
 ?x0 ~films_executive_produced M3 .
 ?x0 ~films_executive_produced M4 .
 ?x1 a film.film
}
```

→ Label: 1 (same)

• Model negative query

```
SELECT DISTINCT ?x0 WHERE {
 ?x0 a people.person .
 ?x0 ~films_executive_produced ?x1 .
 ?x0 ~films_executive_produced M1 .
 ?x0 ~films_executive_produced M2 .
 ?x0 ~films_executive_produced M3 .
 ?x0 ~films_executive_produced M4 .
 ?x1 a film.film
}
```

→ Label: 0 (different)

• Random negative query

```
SELECT count ( * ) WHERE {
 ?x0 a film.cinematographer .
 ?x0 film.editor.film M1 .
 ?x0 film.editor.film M2 .
 ?x0 film.editor.film M3 .
 ?x0 ~films_executive_produced M1 .
 ?x0 ~films_executive_produced M2 .
 ?x0 ~films_executive_produced M3 .
 ?x0 film.writer.film M1 .
 ?x0 film.writer.film M2 .
 ?x0 film.writer.film M3
}
```

→ Label: 0 (different)

(d)

Figure 22: Examples of the CFQ classification dataset. Each query pairs with the question to form an instance. Note the model negative resembles the positive, while the random negative query differs considerably. In the model negative queries, the differences from the positive query are marked in bold.

Symbol	Domain	Description
k	$\mathcal{C}_{\text{in}} \times \mathcal{C}_{\text{in}} \to \mathbb{R}$	Concept kernel (restricted to input concepts)
$\mathbf{x}$	$\mathcal{X}$	A function mapping each input concepts to a values in its domain
$x(c)$	$\mathcal{V}_c$	The function $\mathbf{x}$ evaluated at input concept c
D	$\mathbb{N}$	Number of input concepts
$\mathbf{K}$	$\mathbb{R}^{D\times D}$	Matrix representing the concept kernel (restricted to input concepts)
B	$\mathbb{N}$	Number of channels/bands in each concept domain
b	$\{1,\ldots,B\}$	An index for the b^{th} channel
$\mathbf{x}_b$	$\mathcal{C}_{\text{in}} \to \mathbb{R}$	The function mapping each concept to its value on the b^{th} channel
$\mathbf{M}_{\mathbf{x}}$	$\mathbb{R}^{D\times B}$	The matrix representing the function $\mathbf{x}$, with i^{th} row equal to $x(c)$
m	$\{1,\ldots,D\}$	An index for the m^{th} mode of an eigendecomposition
λ_m	$\mathbb{R}_{\geq 0}$	The m^{th} largest eigenvalue of k
$\boldsymbol{\psi}_m$	$\mathcal{C}_{\text{in}} \to \mathbb{R}$	The m^{th} eigenfunction of k
$\psi_m(c)$	$\mathbb{R}$	The m^{th} eigenfunction of k evaluated at input concept c
α_{bm}	$\mathbb{R}$	The m^{th} Fourier coefficient of the b^{th} channel of $\mathbf{x}$
β_{bm}	$\mathbb{R}$	The m^{th} Fourier coefficient of the b^{th} channel of $\widetilde{\mathbf{x}}$
ξ	$\mathbb{R}_{\geq 0}$	A regularization hyperparameter
$\boldsymbol{\Phi}$	$\mathcal{C}_{\text{in}} \to \mathbb{R}^D$	A concept feature map
$\boldsymbol{\phi}_m$	$\mathcal{C}_{\text{in}} \to \mathbb{R}$	The m^{th} component of the feature map $\boldsymbol{\Phi}$
φ	$\mathbf{x}_b \mapsto \mathbb{R}^D$	A value feature map
$\mathbf{W}^{(l)}_{\{s,t,e\}}$	$\mathbb{R}^{d_{\text{in}}\times d_{\text{out}}}$	Weight matrices in CGAT
$\mathbf{a}^{(l)}$	$\mathbb{R}^{d_{\text{out}}}$	Weight vector in CGAT

Table 22: **Notation.**

Appendix C Concept Kernels for Column Semantics

This appendix provides detailed constructions, algorithms, and extended results underlying Chapter 5, including kernel construction, self-supervised learning pipelines, and additional empirical analyses.

C.1 Notation

Table 22 summarizes the notation used throughout the paper by listing symbols, their mathematical domains, and concise descriptions for easy reference.

C.2 Constructing a Concept Kernel

As described in Section 5.4, concept kernels summarize deterministic information about columns and facilitate the incorporation of domain knowledge into tabular machine learning. The effectiveness of knowledge-enriched tabular machine learning heavily depends on the quality of these kernels, as they dictate the interaction

between the columns. Here, we outline general approaches and key design choices for constructing concept kernels from column metadata.

C.2.1 From concept metadata to concept embeddings

From concept metadata, we extract concept embeddings $(\boldsymbol{\lambda}_c)_{c\in\mathcal{C}}$, where $\mathcal{C}$ is the set of concepts (or columns). Concept embeddings serves the foundation for constructing concept kernels. There are multiple ways to obtain these embeddings[15].

In a case there is a direct mapping from columns to the nodes in a knowledge graph, we can obtain node embeddings using a knowledge graph embedding method [Ji et al., 2021]. However, such direct mappings are often unavailable. For example, in the "Communities and Crime" dataset of KE-TALENT, the columns represent measurements such as "percentage of population that is 12-29 in age", which do not correspond directly to a node in a knowledge graph. A more general approach is to extract sentence embeddings from a language model [Reimers and Gurevych, 2019], using textual descriptions of column names and metadata. This method provides a flexible way to obtain concept embeddings without requiring predefined mappings.

For the datasets in KE-TALENT, either column descriptions or descriptive column names are available in the original data sources. We extract sentence embeddings for each columns to obtain concept embeddings. Categorical columns are preprocessed into one-hot encoding, where each category requires a separate embedding. To generate meaningful presentations, we concatenate the column name with each category label. For example, the category single under the column marital status is converted into marital status is single. When a column description is available, it is appended to the column name before generating embeddings.

The choice of sentence embedding model influences the structure of the concept kernels. You can select a model that performs well across various sentence embedding tasks [Muennighoff et al., 2023], or one specialized for semantic textual similarity (STS) tasks. For sentence embedding models based on instruction tuning, the kernel matrix is often asymmetric. This is because similarity is computed using cosine similarity between query embeddings and document embeddings, with queries being prepended with instructions during embedding extraction. Another consideration is that in many sentence embedding models, similarity scores are more meaningful in terms of relative rankings rather than absolute values. We choose `all-mpnet-base-v2` model due to its widespread adoption and strong performance in general-purpose sentence embedding tasks, as noted in the Sentence-Transformer [Reimers and Gurevych, 2019] documentation.

We note that while our approach is general, there is ample room for improvement by refining column descriptions and selecting a more suitable sentence embedding model through analysis of the concept kernel matrix.

C.2.2 From concept embeddings to concept kernel

Given concept embeddings $(\boldsymbol{\lambda}_c)_{c\in\mathcal{C}}$, we construct concept kernels $k(c, c')$ to provide structured similarity between concepts (or columns). For each dataset, we provide the following concept kernels as baselines:

1. **Inner product:** Here, $k(c, c') = \langle \boldsymbol{\lambda}_c, \boldsymbol{\lambda}_{c'} \rangle$.
2. **Centered inner product:** Let $\overline{\boldsymbol{\lambda}} = \frac{1}{|\mathcal{C}|} \sum_{c\in\mathcal{C}} \boldsymbol{\lambda}_c$. Then $k(c, c') = \langle \boldsymbol{\lambda}_c - \overline{\boldsymbol{\lambda}}, \boldsymbol{\lambda}_{c'} - \overline{\boldsymbol{\lambda}} \rangle$.
3. **Exponential squared distance:** Here, $k(c, c') = \exp\left(-\|\boldsymbol{\lambda}_c - \boldsymbol{\lambda}_{c'}\|_2^2\right)$.

[15] Concept kernels can be directly obtained without explicit concept embeddings, in rare cases where columns have a direct mapping to concepts in a knowledge graph with predefined similarities.

Each of these standard kernels corresponds to a different interpretation of the geometry of the embedding space [Schölkopf, 2002]. Understanding the geometry of sentence embeddings is an active area of research [Park et al., 2024b,a, Templeton, 2024], and such research may pave the way to better ways of constructing kernels for our datasets.

While kernel choice is not the primary focus of our work, we do remark on an important phenomenon. Due to the nature of sentence embeddings, many of these constructions yield kernels which assign high similarity to concept pairs such as $c =$ the-color-is-red and $c' =$ the-color-is-blue. The associated random variables $X(c)$ and $X(c')$ will typically be highly dependent (in terms of mutual information), but they may take on very dissimilar values, e.g., they will tend to have a large *negative* correlation.

Depending on how a method uses the concept kernel, it may be helpful to adapt the kernel to encode such negative relationships. To this end, we provide for each dataset a *concept partition*, i.e. a division of $\mathcal{C}$ into disjoint subsets of concepts $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_E$. From such a partition, we construct an additional kernel:

4. **Group-centered inner product:** Let $\overline{\boldsymbol{\lambda}}_e = \frac{1}{|\mathcal{C}_e|} \sum_{c \in \mathcal{C}_e} \boldsymbol{\lambda}_c$. Then

$$k(c, c') = \begin{cases} \langle \boldsymbol{\lambda}_c - \overline{\boldsymbol{\lambda}}_k, \boldsymbol{\lambda}_{c'} - \overline{\boldsymbol{\lambda}}_k \rangle & c, c' \in \mathcal{C}_k \\ \langle \boldsymbol{\lambda}_c, \boldsymbol{\lambda}_{c'} \rangle & \text{otherwise} \end{cases}$$

The provided concept partitions can also be used for more complex kernel constructions, e.g. using subspace projections inspired by concept algebras [Wang et al., 2024]. Note that, unlike kernels 1-3, this kernel is not necessarily positive definite; possibly limiting what algorithms are applicable.

C.3 Tabular Data Processing

In general, we pre-process tabular data to ensure robust training with ML models. Tabular datasets typically contain two types of columns: numerical (continuous or ordinal values) and categorical (discrete labels). Each type requires appropriate preprocessing to facilitate learning in a deep frameworks, and preprocessing is dependent on the architecture.

Numerical columns can have varying ranges, units and scales (*e.g.*, linear vs. logarithmic). To bring them to a comparable scale while keeping preprocessing simple, we apply column-wise standardization, *i.e.*, subtracting the mean and dividing by the standard deviation. Standardizing numerical features aligns well with extracting spectral components corresponding to smooth graph signals, or eigen functions, and stabilizes training. One can also choose other forms of normalization such as min-max scaling, or, in case the distribution is highly skewed, quantile transformation.

Categorical columns, being nominal, require separate handling. Simply assigning ordinal values to the category labels may introduce arbitrary order to the labels, potentially degrading performance when combined with models that assumes a meaningful order. To avoid this while applying to our various approaches (smoothing, value kernel, and CGAT model), we apply one-hot encoding, representing each category as an independent binary vector. During preprocessing, some of the categorical columns are moved into numerical columns when they represent ordinal values.

C.4 Self-supervised Learning

In this section, we describe the method, implementation details of the pipeline, and key design choices involved in building knowledge-enriched self-supervised learning models for KE-TALENT. The overarching objective of knowledge-enriched SSL model is to learn feature representations from both the raw tabular dataset and the

Algorithm 1 ConceptSSL

Require: Concept kernel $k\colon \mathcal{C} \times \mathcal{C} \to \mathbb{R}$, dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$, losses $(\ell_{\text{SSL}}, \ell_{\text{task}})$, $M \in \mathbb{N}$
Ensure: Encoder $h : \mathcal{X} \to \mathbb{R}^d$, prediction head $g : \mathbb{R}^d \to \mathcal{Y}$

```
## Value transition distribution
Compute the concept transition matrix T and the stationary distribution π_K from k
Compute the value swap distribution Q_{cc'} for each c_i, c_j ∈ C.
Define the value transition distribution ℚ(x'|x) using Equation 14
## Dataset augmentation
Let D_aug = ∅
for m = 1, ..., M do
    Sample x and x' uniformly and independently from D
    Generate augmented views x^1, x^2 ~ ℚ(·|x) and a negative sample x^- ~ ℚ(·|x').
    Add (x^1, x^2, x^-) to D_aug
end for
## Self-supervised learning
Train h on L_SSL(h) = E_{(x^1,x^2,x^-)~D_aug}[ℓ_SSL(f_θ(x^1), f_θ(x^2), f_θ(x^-))]
## Fine-tuning
Train h and g on L_task(h, g) = E_{(x,y)~D}[ℓ_task(g(h(x)), y)]
```

associated concept metadata while leveraging the prior knowledge encoded in concept metadata to improve downstream task performance. Figure 23 illustrates the full pipeline and Algorithm 1 a step-by-step overview of the SSL training procedure.

First, we show how the concept kernel $k\colon \mathcal{C}\times\mathcal{C} \to \mathbb{R}$ can be used to construct a value kernel $K\colon \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ using a self-supervised approach. In the following subsections, we will follow the steps in the figure that are specific to the SSL approach, describing each steps in detail.

Constructing a value transition distribution To begin, assume that $\mathbf{K}_{\text{in}}$ has only nonnegative entries, let $\mathbf{D}$ be a diagonal matrix with $(\mathbf{D})_{ii} = \sum_{j=1}^{d_{\text{in}}} k_{ij}$, and let $\mathbf{T} := \mathbf{D}^{-1}\mathbf{K}_{\text{in}}$. Then $\mathbf{T}$ is a right stochastic matrix, i.e., its rows sum to one. We call $\mathbf{T}$ the *concept transition matrix*, since it defines a *concept Markov chain*, i.e., a Markov chain over concepts. Assuming that $\mathbf{T}$ is irreducible and aperiodic, this Markov chain has a unique stationary distribution which we can represent as a row vector $\boldsymbol{\pi}_{\mathbf{K}}$. With some abuse of notation, we will use $\boldsymbol{\pi}_{\mathbf{K}}$ to denote the stationary distribution, and we will use $\mathbf{T}_i$ to denote the transition distribution from concept c_i.

Now, we can use the concept Markov chain to define a Markov chain over the (potentially uncountable) set of input values $\mathcal{X}$. In particular, given any value $\mathbf{x} \in \mathcal{X}$, consider performing one or more iterations of the following steps:

$$\text{1. Sample } c_i \sim \boldsymbol{\pi}_{\mathbf{K}}. \qquad \text{2. Sample } c_j \sim \mathbf{T}_i. \qquad \text{3. Let } \mathbf{x}^1 \leftarrow \mathsf{swap}(\mathbf{x}\,;\, c_i, c_j).$$

Here, $\mathsf{swap}(\cdot\,;\, c_i, c_j)$ denotes the function which takes $\mathbf{x}$ as input and returns $\mathbf{x}'$ such that $x'(c) = x(c)$ if $c \notin \{c_i, c_j\}$, $x'(c_i) = x(c_j)$, and $x'(c_j) = x(c_i)$. Additionally, to handle (potentially negative) correlation between column values, we further transform the swapped values in $\mathbf{x}^1$ via *value swap distribution* $Q_{cc'}$, a (probabilistic) mapping from $\mathcal{V}_c$ to $\mathcal{V}_{c'}$. This process induces a transition distribution over values, which we call the *value transition distribution*, denoted $\mathbb{Q}(\mathbf{X}^1 \mid \mathbf{X})$:

$$\mathbb{Q}(\mathbf{x}' \mid \mathbf{x}) = \sum_{c,c' \in \mathcal{C}} \boldsymbol{\pi}_{\mathbf{K},c}\mathbf{T}_{cc'}Q_{cc'}(x'_c|x_c). \tag{15}$$

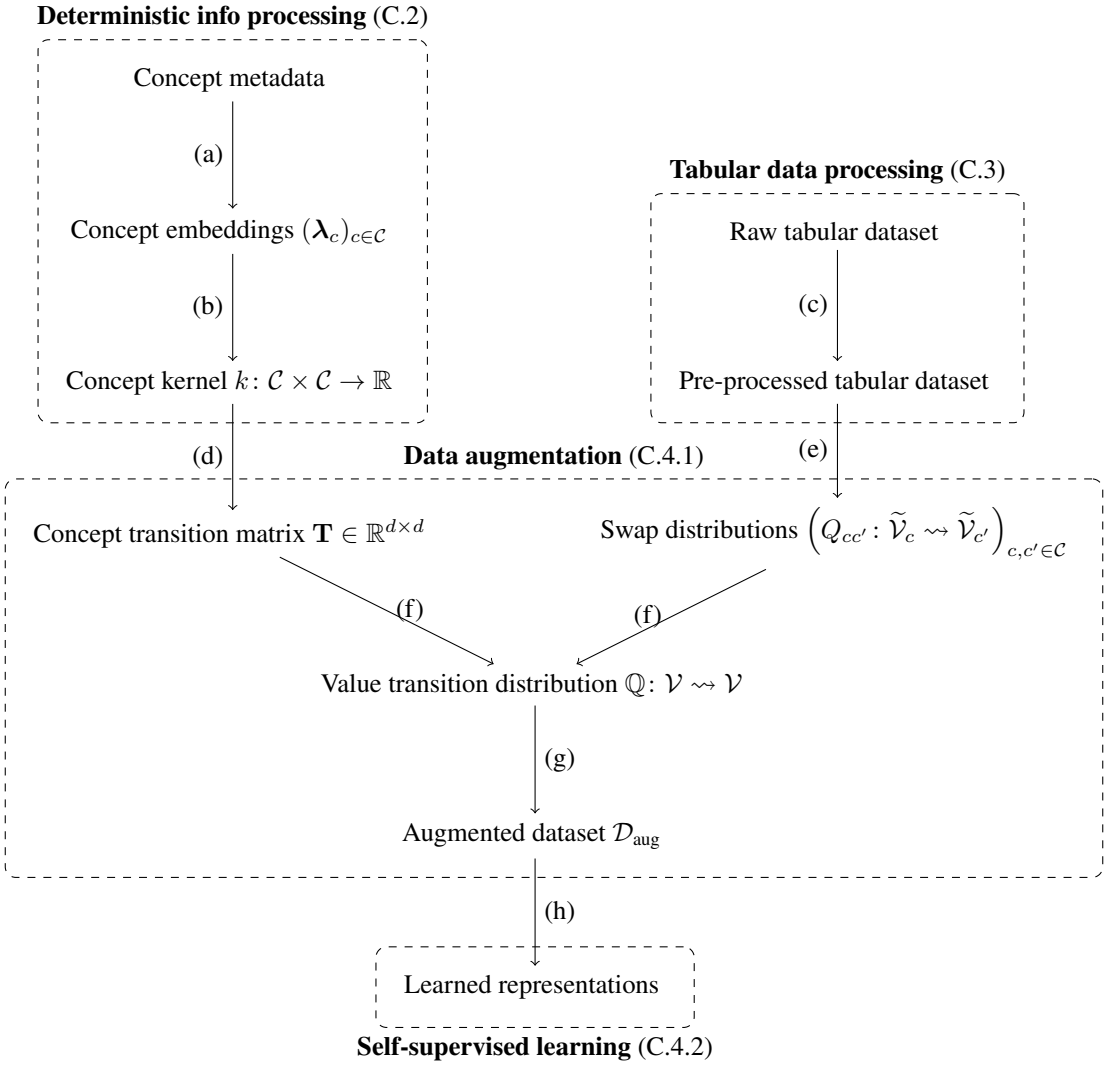


Figure 23: **Full pipeline of self-supervised learning** This figure illustrates the complete pipeline for knowledge-enriched supervised learning. The pipeline consists of multiple stages, including deterministic information processing, tabular data processing, data augmentation, and self-supervised learning. Please refer to the Sections C.2-C.4.2 for the detailed descriptions of each stage.

A related work is Bahri et al. [2022], which augments tabular columns by sampling from marginal distributions, whereas ours leverages concept kernels to incorporate column relationships.

Given the value transition distribution, one can perform augmentation-based self-supervised learning in a number of ways, e.g. for any anchor value $\mathbf{x} \in \mathcal{X}$ taken from our dataset, we can generate a positive sample $\mathbf{x}^1 \sim \mathbb{Q}(\cdot \mid \mathbf{x})$. Then, we can minimize contrastive training objective over these pairs to pre-train a value embedding function $h\colon \mathcal{X} \to \mathcal{Z}$ and use this representation in downstream algorithms. Intriguingly, this process can be seen as transforming each $\mathbf{x}$ by a generalized Fourier expansion in the basis associated with a specific kernel, as we now describe.

The value kernel and eigenspace extraction Given the value transition distribution $\mathbb{Q}(\mathbf{X}^1 \mid \mathbf{X})$ and a data distribution $\mathbb{P}(\mathbf{X})$, we define the value kernel $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as

$$K(\mathbf{x}^1, \mathbf{x}^2) := \frac{\int \mathbb{Q}(\mathbf{x}^1 \mid \mathbf{x}) \cdot \mathbb{Q}(\mathbf{x}^2 \mid \mathbf{x}) \cdot d\mathbb{P}(\mathbf{x})}{\mathbb{P}(\mathbf{x}^1) \cdot \mathbb{P}(\mathbf{x}^2)}, \tag{16}$$

also known as a *positive-pair kernel* [Johnson et al., 2023, Zhai et al., 2024]. Hence, we see that a kernel k over concepts can be used to construct a kernel K over values, which more directly expresses an inductive bias over prediction functions $m\colon \mathcal{X} \to \mathcal{Y}$.

Fortunately, to utilize the value kernel K, we do not need to evaluate it, which would require estimating the data distribution $\mathbb{P}(\mathbf{X})$ and approximating the potentially intractable integral in the numerator. Instead, we may work directly with the eigenspace of this kernel, and there are several potential approaches to efficiently extracting the eigenspace, utilizing its density ratio nature.

A popular class of such approaches is based on *contrastive learning*. For example, HaoChen et al. [2021] define the *spectral contrastive loss*

$$\mathcal{L}_{\text{sc}}(h) := -2 \cdot \mathbb{E}\left[h(\mathbf{X}^1)^\intercal h(\mathbf{X}^2)\right] + \mathbb{E}\left[\left(h(\mathbf{X}^1)^\intercal h(\mathbf{X}^-)\right)^2\right], \tag{17}$$

where the joint distribution over $\left(\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^-\right)$ is given by

$$\int \mathbb{Q}(\mathbf{X}^1 \mid \mathbf{x}) \cdot \mathbb{Q}(\mathbf{X}^2 \mid \mathbf{x}) \cdot \mathbb{Q}(\mathbf{X}^- \mid \mathbf{x}') \cdot d\mathbb{P}(\mathbf{x}) \cdot d\mathbb{P}(\mathbf{x}').$$

They show that, when h ranges over all possible functions, the minimum of Equation (17) yields the true eigenspace of the positive-pair kernel.[16] In practice, h is parameterized by a neural network, and the empirical counterpart of Equation (17) is minimized, thus the eigenspace is not recovered perfectly. See Appendix C.4.2 for other options for self-supervised learning objectives.

C.4.1 Data augmentation

For steps **(a)**-**(c)** of Figure 23, we follow the procedures described in Section C.2 and C.3. In the remainder of this section, we describe the data augmentation and feature representation learning steps specific to the SSL approach.

[16] In particular, this follows from their Lemma 3.2, which relates the spectral contrastive loss to a low-rank approximation of the kernel. See also Table 1 of Johnson et al. [2023].

From concept kernel to concept transition matrix In **(d)**, we transform the concept kernel into the *concept Markov chain* that is needed in the value transition distribution. As described in Section 5.4.4, we row-normalize the concept kernel matrix $\mathbf{K}_{\text{in}}$ to obtain the transition matrix $\mathbf{T}$, from which the stationary distribution $\boldsymbol{\pi}_{\mathbf{K}}$ is derived. The stationary distribution $\boldsymbol{\pi}_{\mathbf{K}}$ specifies the probability of selecting the concept to be swapped, while $\mathbf{T}$ determines the probability to sample to a new concept based on the current one.

To ensure the concept kernels are nonnegative, we clamp negative entries in $\mathbf{K}_{\text{in}}$ to zero. Alternatively, an element-wise transformation, such as exponentiation, can be applied to make all values positive while preserving their relative order. Note that the (cosine) similarity of sentence embeddings, which is the choice of concept kernel in our implementation, is better interpreted by relative order rather than by absolute values.

When selecting concepts to swap using the transition matrix $\mathbf{T}$, we handle numerical and categorical columns differently. When selected concept corresponds to a numerical column, we restrict to choose another numerical columns since we swap the values. In this case, a single transition affects two numerical columns.

For categorical columns, the selected concept represents one of the categories within that column (or the active category of one-hot encoded columns). There are several options for determining the replacement:

- **Swap categories between two categorical columns**: Choose another categorical column and exchange their active categories.
- **Replace the category within the same column**: Select a different category from the same column and replace the current category.
- **Replace the category with one from any categorical column**: Choose a category from any categorical column and substitute it for the current one.

Note the first option modifies two categorical columns, while the latter two affect only a single column.

Learning swap distributions In **(e)**, we define how the actual swapping of column values occurs after columns are selected by the concept transition matrix. If we simply interchange the values between two columns, the process becomes equivalent to directly using the concept kernels as the correlation of column values. However, there are potentially more sophisticated methods that we can apply to better model the swap distribution $Q_{cc'} : \widetilde{\mathcal{V}}_c \rightsquigarrow \widetilde{\mathcal{V}}_{c'}$ as a probabilistic mapping.

For numerical columns, one straightforward approach to model $Q_{cc'}$ is to simply apply a linear transformation to the swapped column values. The weight and bias of this transformation can be pre-determined or trained through self-supervised learning. Possible strategies include:

- **Linear regression**: Fix the weights and bias to the solution of the linear regression. Under standardization, the weight corresponds to the correlation $\rho_{cc'}$ and the bias is set to zero.
- **Correlation sign**: Fix the weight to be the sign of the correlation and the bias to zero. This addresses the problem of negative value correlation between two close concepts.
- **Trainable weights**: Initialize the weight and bias to those from linear regression and allow them to be updated during training.

We chose the first option as the swap distribution on numerical columns. However, one can model $Q_{cc'}$ in a more complex manner, such as incorporating trainable weights or making it probabilistic by predictive posterior variance.

Since we adopt FT-Transformer Gorishniy et al. [2021] as the feature encoder, each category in categorical columns is represented as an embedding vector. Transition of categorical columns corresponds to changing one embeddings to another. For simplicity and to keep the number of trainable parameters small, we use the changed embedding directly without further transformations.

Constructing the value transition distribution In **(f)**, we construct the value transition distribution $\mathbb{Q}\colon \mathcal{V} \rightsquigarrow \mathcal{V}$ by integrating the concept transition distribution $\mathbf{T}$ with the value swap distribution $Q_{cc'}$. Specifically, given a concept c sampled from the stationary distribution $\pi_{\mathbf{K}}$ and its paired transition concept $c' \sim \mathbf{T}_c$, the transition of values follows $Q_{cc'}(x'_c|x_c)$. Formally, the value transition distribution is defined as

$$\mathbb{Q}(\mathbf{x}'|\mathbf{x}) = \sum_{c,c' \in \mathcal{C}} \pi_{\mathbf{K},c} \mathbf{T}_{cc'} Q_{cc'}(x'_c|x_c).$$

Data augmentation In **(g)**, we construct the augmented dataset $\mathcal{D}_{\text{aug}}$, we apply the value transition distribution $\mathbb{Q}$ to generate augmented views of samples in $\mathcal{D}$. By applying $\mathbb{Q}$ to a given input instance (or row) $\mathbf{x}$, we obtain a transformed sample $\mathbf{x}'$, which preserves the semantics while introducing controlled perturbations. In Section C.4.1, $\mathcal{D}_{\text{aug}}$ formulates a set of triplets $(\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^-)$, where $\mathbf{x}^1$ and $\mathbf{x}^2$ are positive pairs and $\mathbf{x}^-$ is a negative sample, as structure particularly useful for spectral contrastive loss and other triplet-based objectives. However, for InfoNCE loss where negative samples are implicitly drawn from the batch rather than explicitly drawn from the mini batch, so different formulation of $\mathcal{D}_{\text{aug}}$ is required.

For datasets with many columns, we allow multiple value transitions per sample, where the number of transitions is selected based on the number of columns. Appendix C.5 describes the heuristics in determining the number of transitions per different dataset to ensure sufficient augmentation. Also, note that additional stochasticity can be incorporated into $\mathcal{D}_{\text{aug}}$ to enhance diversity, such as introducing noise into numerical columns.

C.4.2 Self-supervised representation learning

So far, we have obtained the augmented dataset $\mathcal{D}_{\text{aug}}$ from value transition distribution. In **(h)**, using this dataset, we perform contrastive representation learning to learn feature representation of instances and to perform supervised learning. The training process consist of two main steps:

- **Contrastive learning to learn features**: The feature encoder $h(\mathbf{x})$ is initially trained on $\mathcal{D}_{\text{aug}}$ using a contrastive learning objective to extract meaningful representation.
- **Fine-tuning (or transfer learning)**: The trained encoder is then fine-tuned with a prediction head for the supervised task dataset $\mathcal{D}$.

The choice feature encoder $h(\mathbf{x})$ can be arbitrary as long as it produces a fixed-length feature from tabular row. Therefore, various architectures can be employed. As described in Section 5.5.1, we choose FT-Transformer as our main encoder due to its input encoding scheme (feature tokenizer). Alternatively, GNN-based encoders or simple fully-connected networks can be adapted depending on the dataset of interest.

Several learning objectives can be used for SSL [Johnson et al., 2023, Bardes et al., 2022]:

- **Spectral contrastive loss**: As described above, this approach leverages spectral properties of the data.
- **NT-Xent/NT-Logistic**: These is a widely used class of objectives in contrastive learning that maximizes similarity between augmented views of the same instance while minimizing similarity with negative samples.
- **Non-contrastive losses**: One can factorize the positive-pair kernel using *non-contrastive* objectives, which extend the Rayleigh quotient to this stochastic setting [Horn and Johnson, 2012]. A popular

instance is the VICReg objective [Bardes et al., 2022], which is a Lagrangian objective towards solving:

$$\begin{aligned} \mathcal{L}_{\text{non-con}}(h) := \mathbb{E}\left\|h(\mathbf{X}^1) - h(\mathbf{X}^2)\right\|_2^2 \\ \text{s.t. Cov}(h(\mathbf{X}^i)) = I,\ i = 1, 2, \end{aligned} \tag{18}$$

where the joint distribution over $\mathbf{X}^1$ and $\mathbf{X}^2$ is given by

$$\int \mathbb{Q}(\mathbf{X}^1 \mid \mathbf{x}) \cdot \mathbb{Q}(\mathbf{X}^2 \mid \mathbf{x}) \cdot d\mathbb{P}(\mathbf{x}),$$

i.e., the distribution induced by first sampling $\mathbf{x}$ from the data distribution, then sampling $\mathbf{X}^1$ and $\mathbf{X}^2$ from the value transition distribution conditioned on $\overline{\mathbf{x}}$.

From several experiments, we found that the InfoNCE loss [Oord et al., 2018] which falls under NT-Xent, shows better performance than the spectral contrastive loss.

Similar to the benchmark method in Ye et al. [2024a], we perform hyper-parameter tuning to benchmark our methods. For our self-supervised learning method in Section 5.4.4, the training of our knowledge-enriched model occurs in two steps, so we searched for hyper-parameters only in the second step. Please refer to Section C.5 for details.

C.5 Hyper-parameter Search and Training Details

To optimize our methods to each dataset of KE-TALENT benchmark, we conducted hyper-parameter optimization using Optuna Akiba et al. [2019] with the Tree-structured Parzan Estimator (TPE) sampler, the default search strategy in Optuna, following Ye et al. [2024a]. The search process aimed to optimize the root mean squared error (RMSE) for regression tasks and maximize the accuracy for classification tasks on the validation set. After finding the optimal hyper-parameters for each dataset and model, we trained the model 15 times with different random seeds and reported the test set performance. The search space included both architectural and training parameters as detailed in Table 23 for the hyper-parameter search space.

For the smoothing and value kernel models, which use RealMLP as their MLP architecture, we searched over the same hyper-parameter space as the TALENT benchmark, except for narrowing the range of the learning rate to prevent instability in training. We used the AdamW optimizer with $\beta_1 = 0.9, \beta_2 = 0.95$, a batch size of 256, and a coslog_4 learning rate scheduler. In our implementation of RealMLP, we omitted data-driven weight initialization or a decaying dropout ratio due to the implementation complexity. Note that, despite these simplifications, our version remains closer to the original RealMLP compared to its simplified variant, RealMLP-TD-S.

For the concept graph attention network (CGAT) model, we used dataset-specific batch sizes. Specifically, we use the largest possible batch size (a power of two, up to 1024) for each dataset that could fit within 48GB of VRAM (NVIDIA RTX A6000 GPU).

For the self-supervised learning (SSL) model, hyper-parameter search was performed only during the fine-tuning stage, as described in Appendix C.4.2, focusing on the MLP prediction head and optimizer settings. During the SSL stage, we followed the TALENT codebase for default hyper-parameters of the encoder (FT-Transformer) architecture. We employed the AdamW optimizer with a cosine learning rate scheduler with warmup, where the base learning rate was set to 2×10^{-5}, and no weight decay. The model was trained with a batch size of 1024, and the number of epochs was chosen as the minimum multiple of 100 that ensured training for 1000 steps. The number of transition steps during data augmentation was adjusted per dataset to ensure that approximately 20% of columns were modified in each augmentation step. For the InfoNCE loss, we set the temperature parameter to $\tau = 0.1$.

Methods	Parameters	Grid
Smoothing	Smoothing ξ (Only for norm and Laplacian)	LogUniform [0.1, 10.0]
	Numerical embedding	{none, pbld}
	Dropout	{0.0, 0.15}
	Nonlinear activation	{SELU, Mish}
	MLP hidden layers	{[256,256,256],[64,64,64,64,64],[512]}
	PLR sigma	LogUniform [0.05, 0.5]
	Label smoothing (only for classification)	{0.0, 0.1}
	Learning rate	LogUniform [0.02, 0.07]
	Weight decay	{0.0, 0.02}
Value kernel	Spectral decomposition matrix	{Adjacency, Laplacian}
	Numerical embedding	{none, pbld}
	Dropout	{0.0, 0.15}
	Nonlinear activation	{SELU, Mish}
	MLP hidden layers	{[256,256,256],[64,64,64,64,64],[512]}
	PLR sigma	LogUniform [0.05, 0.5]
	Label smoothing (only for classification)	{0.0, 0.1}
	Learning rate	LogUniform [0.02, 0.07]
	Weight decay	{0.0, 0.02}
Concept graph attention networks (CGAT)	Input embed dim	LogInt [16, 256]
	Conv num layers	UniformInt [1, 3]
	Conv hidden dim	LogInt [16, 64]
	Concept attention dim	LogInt [4, 16]
	Num attn heads	UniformInt [1, 4]
	Edge active ratio	Uniform [0.1, 0.9]
	MLP num layers	UniformInt [2, 5]
	MLP hidden dim	LogInt [16, 256]
	MLP dropout	0.1
	Learning rate	LogUniform [3e-5, 1e-3]
	Weight decay	LogUniform [1e-6, 1e-3]
SSL	MLP num layers	UniformInt [2, 5]
	MLP hidden dim	LogInt [16, 256]
	MLP dropout	0.1
	Learning rate	LogUniform [3e-5, 3e-3]
	Weight decay	LogUniform [1e-6, 1e-3]

Table 23: Hyper-parameter space for knowledge-enriched supervised learning methods

C.6 Additional Results

C.7 Best result for each dataset

To provide a more complete picture, Table 24 lists extended KE-TALENT results, including the single best score for every dataset in the original TALENT benchmark. We did not treat these methods as primary

baselines because (i) they were not the overall top performer within their respective model families, or (ii) their results became public only after our paper was written.

C.7.1 Concept kernel visualization

In Figures 24 to 26, we visualize the concept kernels for three datasets with 20–34 columns: Student Performance, German Credit Data, and Student Dropout. Additionally, Tables 25 to 27, lists the top-5 nearest columns (highest cosine similarity in the concept embedding space, excluding self) for each column in those datasets. Many of these neighbor pairs are indeed semantically related. For example:

- **Student Performance**
  - "number of school absences" and "number of past class failures"
  - "weekend alcohol consumption" and "workday alcohol consumption"
  - "extra educational support" and "family educational support"
  - "mother's education" and "father's education"
- **German Credit Data**
  - "Credit amount" and "Number of existing credits at this bank"
  - "Present residence since" and "Present employment since"
  - "Housing" and "Property"
  - "Other installment plans" and "Installment rate in percentage of disposable income"
- **Student Dropout**
  - "Mother's qualification" and "Father's qualification"
  - "Previous qualification" and "Previous qualification (grade)"
  - "Nationality" and "International"
  - "Mother's occupation" and "Father's occupation"

Because the concept embeddings were generated by a pretrained language model, these findings further demonstrate that modern LMs are effective at computing concept kernels that capture meaningful semantic relationships among dataset columns.

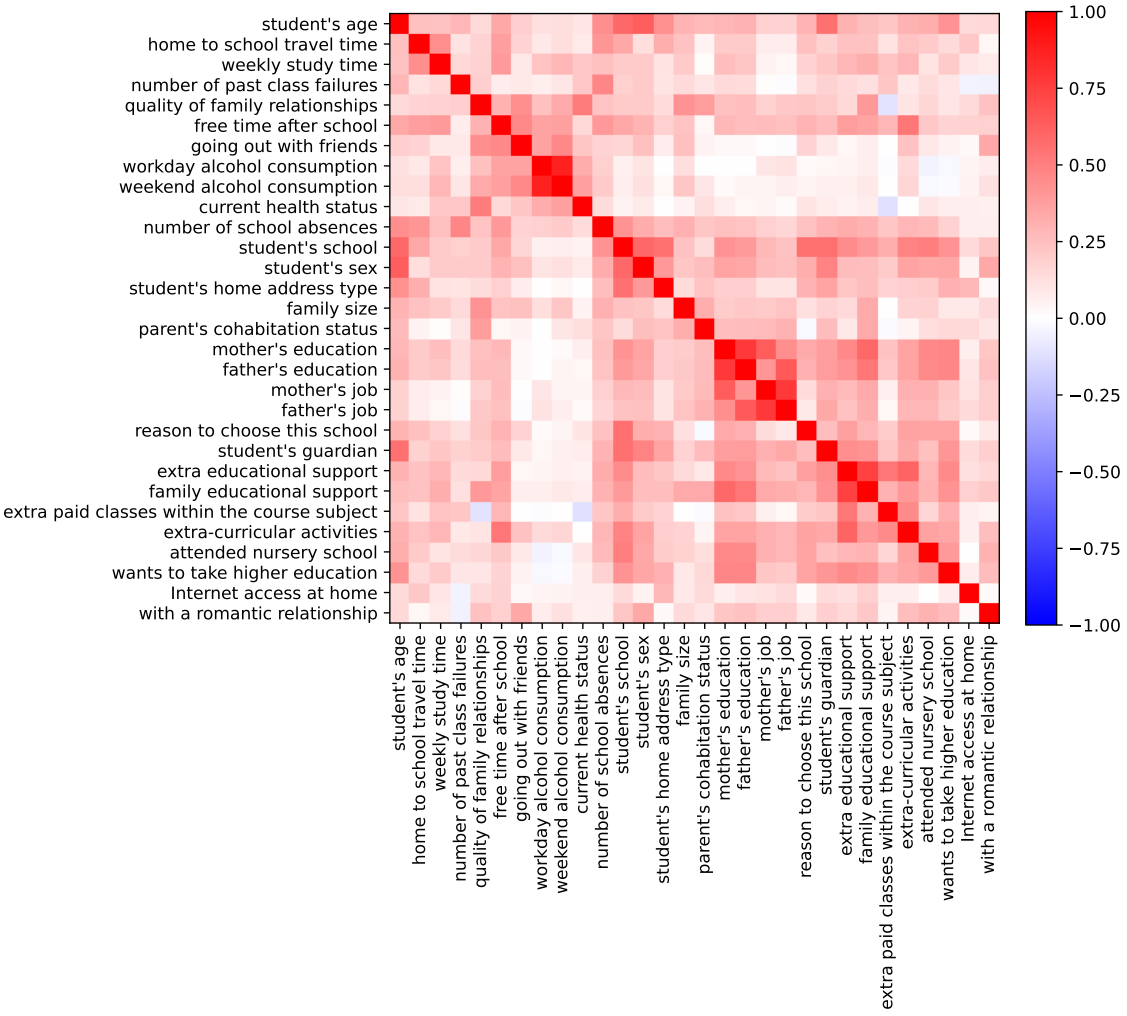


Figure 24: Kernel heatmap of Student Performance dataset

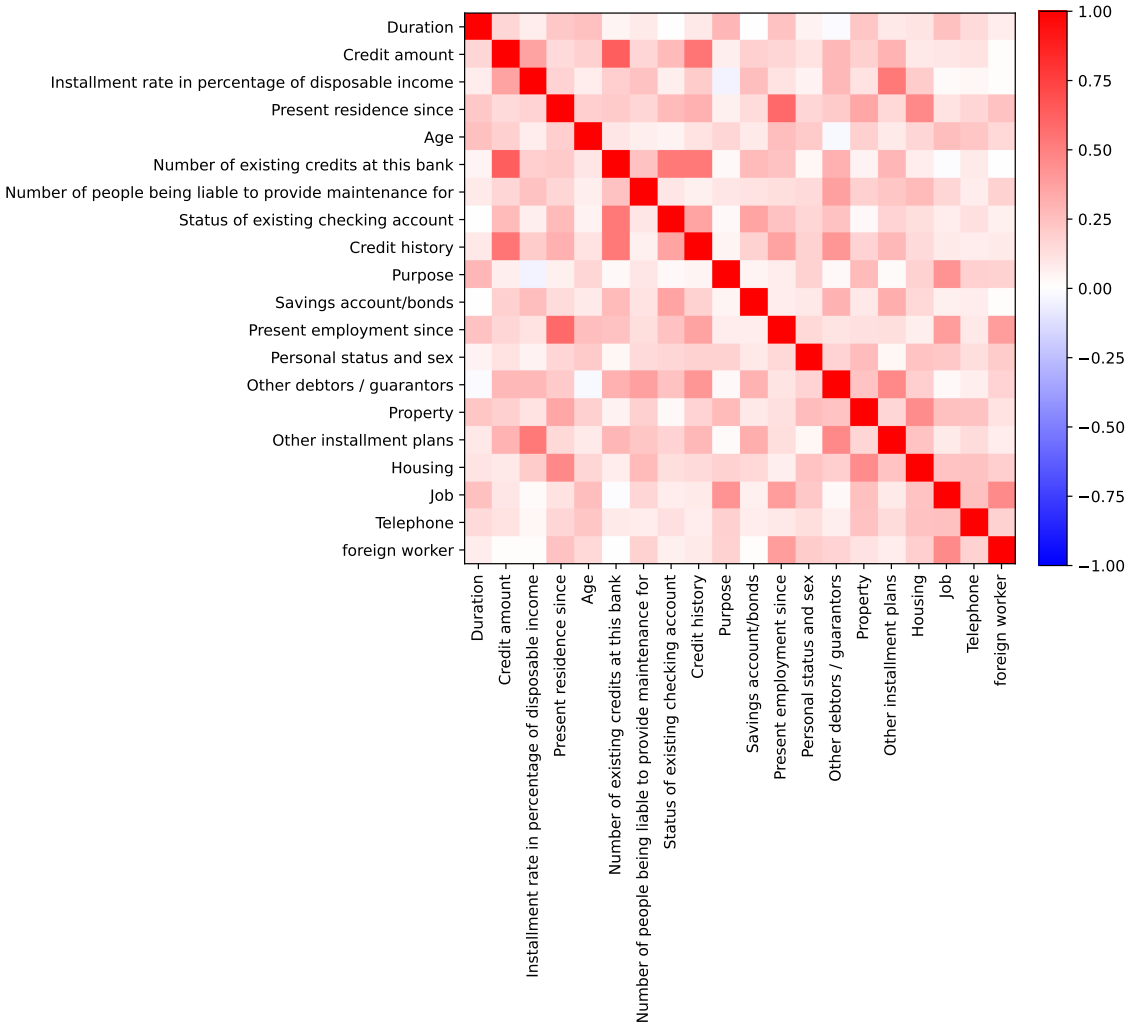


Figure 25: Kernel heatmap of German Credit Data dataset

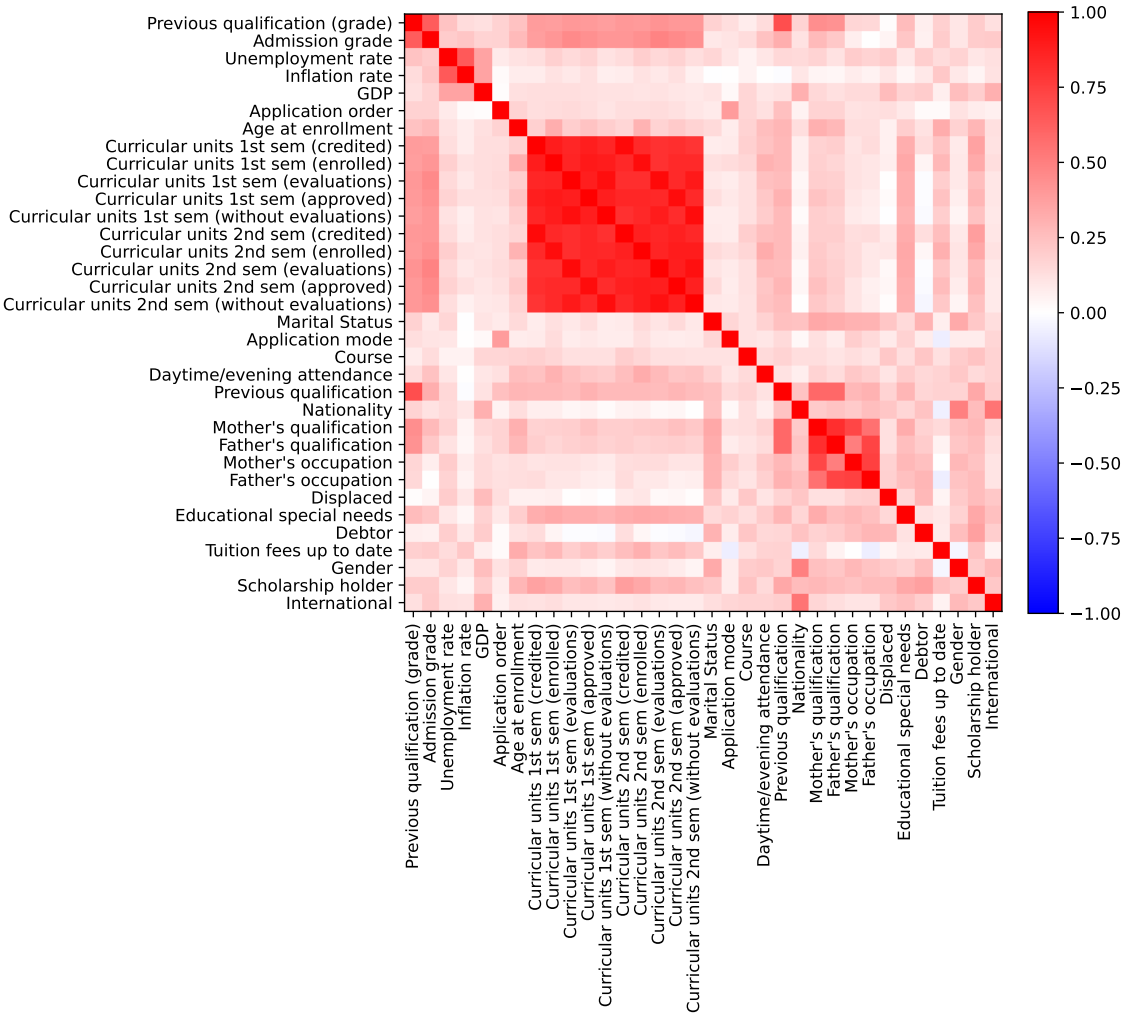


Figure 26: Kernel heatmap of Student Dropout dataset

Dataset	**Abalone**	**Diamond**	**ParkTel**	**StuPerf**	**Crime**	**Churn**	**Credit**	**Taiwan**	**ASP**	**Internet**	**StuDrop**
Method \ Task	reg/RMSE(↓)					bincls/Acc(↑)			multicls/Acc(↑)		
RealMLP	2.1210	523.92	**0.7337**	2.9277	0.1381	0.8735	0.7157	0.9667	0.3861	0.5302	0.7655
CatBoost	2.1789	524.91	1.5994	2.9244	<u>0.1336</u>	<u>0.8759</u>	<u>0.7430</u>	<u>0.9718</u>	0.3815	**0.5358**	**0.7782**
TabR	2.1078	513.53	8.0521	2.9072	0.1437	0.8743	0.7240	0.9678	0.3750	0.5183	0.7493
FT-T	2.1078	532.83	8.3437	2.9642	0.1369	0.8709	0.7123	0.9674	0.3678	<u>0.5348</u>	0.7547
Best	2.0888 (ResNet)	**492.58** (MNCA)		**2.5560** (MNCA)	**0.1325** (XGB)		**0.7513** (LGBM)		**0.4448** (XGB)		
Smooth(kernel)	2.1718	938.03	2.4700	3.0651	0.1466	0.8657	0.7160	**0.9722**	0.3815	0.5042	0.7162
Smooth(norm)	2.0879	903.70	1.2112	2.9725	0.1401	0.8688	0.6960	0.9659	0.4013	0.5093	0.7579
Smooth(Laplacian)	2.0937	522.37	0.9530	2.8926	0.1397	**0.8765**	0.7193	0.9694	0.3900	0.5259	0.7673
Value kernel	**2.0825**	525.79	0.8676	2.9203	0.1394	<u>0.8746</u>	0.7157	0.9673	0.3761	0.5315	0.7665
CGAT	2.0876	677.33	1.4612	3.0397	0.1425	<u>0.8764</u>	0.7337	0.9675	0.3838	0.5275	0.7656
SSL	2.1584	534.12	1.1275	2.9178	0.1373	<u>0.8757</u>	0.7180	0.9655	0.3964	<u>0.5327</u>	0.7571

Table 24: **Additional Results on KE-TALENT benchmark** Additionally to Table 12, the table reports the best method for each dataset in the "Best" row in case if the earlier four baselines didn't achieve the best in TALENT (MNCA: ModernNCA [Ye et al., 2024b], XGB: XGBoost [Chen and Guestrin, 2016], LGBM: LightGBM [Ke et al., 2017]).

Column	Top-5 Nearest Columns
“student’s age”	“student’s sex” “student’s school” “student’s guardian” “number of school absences” “student’s home address type”
“home to school travel time”	“weekly study time” “number of school absences” “free time after school” “student’s school” “student’s home address type”
“weekly study time”	“home to school travel time” “free time after school” “family educational support” “extra-curricular activities” “extra educational support”
“number of past class failures”	“number of school absences” “student’s age” “extra paid classes within the course subject” “current health status” “student’s sex”
“quality of family relationships”	“current health status” “going out with friends” “family size” “family educational support” “parent’s cohabitation status”
“free time after school”	“extra-curricular activities” “going out with friends” “number of school absences” “weekly study time” “extra educational support”
“going out with friends”	“weekend alcohol consumption” “free time after school” “quality of family relationships” “workday alcohol consumption” “with a romantic relationship”
“workday alcohol consumption”	“weekend alcohol consumption” “going out with friends” “free time after school” “current health status” “quality of family relationships”
“weekend alcohol consumption”	“workday alcohol consumption” “going out with friends” “current health status” “free time after school” “quality of family relationships”
“current health status”	“quality of family relationships” “weekend alcohol consumption” “workday alcohol consumption” “going out with friends” “weekly study time”

Column	Top-5 Nearest Columns
“number of school absences”	“number of past class failures” “student’s age” “student’s school” “home to school travel time” “free time after school”
“student’s school”	“student’s sex” “student’s age” “student’s guardian” “reason to choose this school” “student’s home address type”
“student’s sex”	“student’s age” “student’s school” “student’s guardian” “student’s home address type” “extra-curricular activities”
“student’s home address type”	“student’s school” “student’s age” “student’s sex” “student’s guardian” “home to school travel time”
“family size”	“quality of family relationships” “family educational support” “parent’s cohabitation status” “number of school absences” “student’s age”
“parent’s cohabitation status”	“quality of family relationships” “family educational support” “family size” “father’s job” “student’s age”
“mother’s education”	“father’s education” “mother’s job” “family educational support” “wants to take higher education” “attended nursery school”
“father’s education”	“mother’s education” “father’s job” “family educational support” “wants to take higher education” “attended nursery school”
“mother’s job”	“father’s job” “mother’s education” “father’s education” “family educational support” “student’s guardian”
“father’s job”	“mother’s job” “father’s education” “mother’s education” “student’s guardian” “family educational support”

Column	Top-5 Nearest Columns
“reason to choose this school”	“student’s school” “extra educational support” “extra-curricular activities” “attended nursery school” “wants to take higher education”
“student’s guardian”	“student’s school” “student’s age” “student’s sex” “extra educational support” “family educational support”
“extra educational support”	“family educational support” “extra-curricular activities” “extra paid classes within the course subject” “mother’s education” “wants to take higher education”
“family educational support”	“extra educational support” “mother’s education” “father’s education” “wants to take higher education” “student’s guardian”
“extra paid classes within the course subject”	“extra educational support” “extra-curricular activities” “student’s school” “wants to take higher education” “family educational support”
“extra-curricular activities”	“extra educational support” “free time after school” “student’s school” “extra paid classes within the course subject” “family educational support”
“attended nursery school”	“student’s school” “mother’s education” “father’s education” “wants to take higher education” “extra-curricular activities”
“wants to take higher education”	“mother’s education” “father’s education” “extra educational support” “student’s age” “family educational support”
“Internet access at home”	“student’s home address type” “home to school travel time” “family educational support” “free time after school” “student’s school”
“with a romantic relationship”	“student’s sex” “going out with friends” “attended nursery school” “wants to take higher education” “extra-curricular activities”

Table 25: **Top-5 nearest columns for each column in Student Performance dataset**

Column	Top-5 Nearest Columns
“Duration”	“Purpose” “Job” “Age” “Present employment since” “Property”
“Credit amount”	“Number of existing credits at this bank” “Credit history” “Installment rate in percentage of disposable income” “Other installment plans” “Other debtors / guarantors”
“Installment rate in percentage of disposable income”	“Other installment plans” “Credit amount” “Other debtors / guarantors” “Savings account/bonds” “Number of people being liable to provide maintenance for”
“Present residence since”	“Present employment since” “Housing” “Property” “Credit history” “Status of existing checking account”
“Age”	“Job” “Present employment since” “Duration” “Telephone” “Personal status and sex”
“Number of existing credits at this bank”	“Credit amount” “Credit history” “Status of existing checking account” “Other debtors / guarantors” “Other installment plans”

Column	Top-5 Nearest Columns
“Number of people being liable to provide maintenance for”	“Other debtors / guarantors” “Housing” “Number of existing credits at this bank” “Installment rate in percentage of disposable income” “Other installment plans”
“Status of existing checking account”	“Number of existing credits at this bank” “Savings account/bonds” “Credit history” “Credit amount” “Present residence since”
“Credit history”	“Credit amount” “Number of existing credits at this bank” “Other debtors / guarantors” “Present employment since” “Status of existing checking account”
“Purpose”	“Job” “Duration” “Property” “Telephone” “foreign worker”
“Savings account/bonds”	“Status of existing checking account” “Other installment plans” “Other debtors / guarantors” “Number of existing credits at this bank” “Installment rate in percentage of disposable income”
“Present employment since”	“Present residence since” “foreign worker” “Job” “Credit history” “Age”
“Personal status and sex”	“Property” “Housing” “Job” “Age” “foreign worker”

Column	Top-5 Nearest Columns
“Other debtors / guarantors”	“Other installment plans” “Credit history” “Number of people being liable to provide maintenance for” “Number of existing credits at this bank” “Savings account/bonds”
“Property”	“Housing” “Present residence since” “Purpose” “Personal status and sex” “Job”
“Other installment plans”	“Installment rate in percentage of disposable income” “Other debtors / guarantors” “Savings account/bonds” “Credit amount” “Number of existing credits at this bank”
“Housing”	“Present residence since” “Property” “Number of people being liable to provide maintenance for” “Telephone” “Personal status and sex”
“Job”	“foreign worker” “Purpose” “Present employment since” “Age” “Duration”
“Telephone”	“Job” “Housing” “Property” “Age” “Purpose”
“foreign worker”	“Job” “Present employment since” “Present residence since” “Personal status and sex” “Housing”

Table 26: **Top-5 nearest columns for each column in German Credit Data dataset**

Column	Top-5 Nearest Columns
“Previous qualification (grade)”	“Previous qualification” “Admission grade” “Mother’s qualification” “Father’s qualification” “Curricular units 2nd sem (evaluations)”
“Admission grade”	“Previous qualification (grade)” “Curricular units 2nd sem (evaluations)” “Curricular units 1st sem (evaluations)” “Curricular units 2nd sem (approved)” “Curricular units 2nd sem (without evaluations)”
“Unemployment rate”	“Inflation rate” “GDP” “Previous qualification (grade)” “Mother’s occupation” “Admission grade”
“Inflation rate”	“Unemployment rate” “GDP” “Admission grade” “Tuition fees up to date” “Previous qualification (grade)”
“GDP”	“Unemployment rate” “Inflation rate” “Nationality” “International” “Displaced”
“Application order”	“Application mode” “Previous qualification” “Previous qualification (grade)” “Admission grade” “Father’s qualification”
“Age at enrollment”	“Tuition fees up to date” “Curricular units 1st sem (enrolled)” “Mother’s qualification” “Curricular units 2nd sem (enrolled)” “Previous qualification”
“Curricular units 1st sem (credited)”	“Curricular units 2nd sem (credited)” “Curricular units 1st sem (enrolled)” “Curricular units 1st sem (approved)” “Curricular units 1st sem (evaluations)” “Curricular units 1st sem (without evaluations)”
...	

Column	Top-5 Nearest Columns
...	
“Curricular units 2nd sem (without evaluations)”	“Curricular units 1st sem (without evaluations)” “Curricular units 2nd sem (evaluations)” “Curricular units 1st sem (evaluations)” “Curricular units 2nd sem (enrolled)” “Curricular units 2nd sem (approved)”
“Marital Status”	“Mother’s qualification” “Gender” “Father’s qualification” “Mother’s occupation” “Debtor”
“Application mode”	“Application order” “Mother’s occupation” “Educational special needs” “International” “Marital Status”
“Course”	“Scholarship holder” “Debtor” “Displaced” “Gender” “Curricular units 2nd sem (credited)”
“Daytime/evening attendance”	“Curricular units 2nd sem (enrolled)” “Curricular units 1st sem (enrolled)” “Curricular units 2nd sem (evaluations)” “Curricular units 2nd sem (credited)” “Age at enrollment”
“Previous qualification”	“Previous qualification (grade)” “Mother’s qualification” “Father’s qualification” “Scholarship holder” “Admission grade”
“Nationality”	“International” “Gender” “GDP” “Scholarship holder” “Father’s occupation”
“Mother’s qualification”	“Father’s qualification” “Mother’s occupation” “Previous qualification” “Father’s occupation” “Previous qualification (grade)”
“Father’s qualification”	“Mother’s qualification” “Father’s occupation” “Previous qualification” “Mother’s occupation” “Previous qualification (grade)”

Column	Top-5 Nearest Columns
“Mother’s occupation”	“Father’s occupation” “Mother’s qualification” “Father’s qualification” “Marital Status” “Previous qualification”
“Father’s occupation”	“Mother’s occupation” “Father’s qualification” “Mother’s qualification” “Previous qualification” “Marital Status”
“Displaced”	“Debtor” “Scholarship holder” “GDP” “Marital Status” “Nationality”
“Educational special needs”	“Curricular units 2nd sem (enrolled)” “Scholarship holder” “Curricular units 1st sem (enrolled)” “Curricular units 2nd sem (evaluations)” “Mother’s qualification”
“Debtor”	“Scholarship holder” “Marital Status” “Father’s occupation” “Displaced” “Gender”
“Tuition fees up to date”	“Age at enrollment” “Curricular units 2nd sem (enrolled)” “Curricular units 1st sem (enrolled)” “Curricular units 2nd sem (approved)” “Curricular units 2nd sem (credited)”
“Gender”	“Nationality” “Marital Status” “Mother’s occupation” “International” “Debtor”
“Scholarship holder”	“Curricular units 2nd sem (credited)” “Debtor” “Curricular units 1st sem (credited)” “Curricular units 2nd sem (enrolled)” “Educational special needs”
“International”	“Nationality” “GDP” “Gender” “Educational special needs” “Displaced”

Table 27: **Top-5 nearest columns for each column in Student Dropout dataset** We omitted several redundant columns (from “Curricular units 1st sem (enrolled)” to “Curricular units 2nd sem (approved)”) due to space constraints.