Word Sense Disambiguation Using Semi-Supervised Naive Bayes with Ontological Constraints

Jakob Bauer

Wednesday 23rd November, 2016

Abstract

Background. Word sense disambiguation (WSD) is the task of mapping an ambiguous word to its correct sense given its context. As high-quality sense-tagged data is scarce and expensive to obtain, attention has shifted from fully-supervised to semi-supervised and knowledge-based approaches to WSD that rely on a lexical knowledge base such as WordNet instead of large amounts of hand-labeled data. What is currently missing is a method to effectively combine elements of semi-supervised and knowledge-based approaches into a single system.

Aim. Our goal is to improve the performance of semi-supervised and knowledge-based lexical-sample WSD on a benchmark dataset by designing a system that uses both automatically acquired examples and explicitly models ontological constraints between senses in the classification stage.

Data. Our training data consists of automatically acquired examples unlabeled from the ukWaC corpus, and glosses and example usages from WordNet. The system is evaluated on twelve different target lemmas from the Koeling et al. (2005) benchmark dataset.

Methods. We use a semi-supervised Naive Bayes classifier that is trained on automatically acquired examples and that explicitly takes into account ontological constraints between senses at the classification stage.

Results. We find that our method does not uniformly outperform state-of-the-art baselines such as gloss vectors and personalized PageRank. Possible reasons are semantic drift, deficiencies in how the ontological constraints are modeled, and bad sense priors.

Conclusions. Although our current system is not able to outperform state-of-the-art baselines, we believe that the error analysis provided in this paper can help guide future research in effectively combining semi-supervised and knowledge-based approaches to WSD.

Keywords: Word sense disambiguation, WSD

1 Introduction

Word sense disambiguation (WSD) is the task of mapping an ambiguous word to its correct meaning given its context. For instance, we might want to disambiguate the word "goal" in the sentence: "Italian striker Mario Balotelli scores a goal against Monaco." Among the different candidate senses for goal (see Figure 2 in the appendix), "a successful attempt at scoring" is the most appropriate one.

WSD is an important and challenging task: important because it is a stepping stone to more advanced natural language processing tasks such as machine translation and question answering, and challenging because even after several decades of research, it is still a largely unsolved problem.

The most successful approaches to WSD to date are fully supervised systems that are trained on sense-tagged corpora (Navigli, 2009). Unfortunately, manually tagging data is a slow and expensive process, which means that it is all but impossible to generate sufficiently large hand-tagged corpora outside of a few particular domains. As a result, supervised WSD is limited in its scope. The WSD community has tried to work around this problem by shifting its attention to semi-supervised and knowledge-based disambiguation methods that only require minimal supervision in form of a lexical knowledge base such as WordNet (Miller, 1995).

One successful semi-supervised approach is based on monosemous relatives, i.e., unambiguous words that WordNet considers to be closely related to the target word. After the monosemous relatives have been identified, they can be used to acquire labeled training examples which in turn can be used to train a supervised classifier (Martinez et al., 2008). An example of a successful knowledge-based approach is the use of the personalized PageRank algorithm on a graph that uses WordNet synsets as nodes and semantic relationships between synsets as edges. Probability mass is injected into the graph according to the context of the word that is being disambiguated (Agirre and Soroa, 2009).

Our approach tries to combine ideas from both semi-supervised and knowledge-based WSD. In particular, we use a WordNet ontology built around all possible senses of the target word to first automatically acquire labeled training examples. We then combine these examples with unlabeled examples to train a semi-supervised Naive Bayes classifier using the expectation maximization (EM) technique. During the E-step of EM, the classifier also penalizes sense assignments that violate ontological constraints between senses such as *subset-of* and *mutex* (Dalvi et al., 2015).

We believe that our approach could be of interest to the WSD community because it shows a new way in which elements from semi-supervised and knowledge-based approaches can be combined into a single approach.

2 Problem and Approach

The problem is to uniformly increase macro-averaged recall in lexical-sample WSD on selected lemmas of the Koeling et al. (2005) benchmark dataset using a semi-supervised Naive Bayes classifier that explicitly takes into account ontological constraints between senses at the classification stage.

3 Background and Related Work

In this section, we will first review a hierarchical semi-supervised approach to another task called gloss finding (Dalvi et al., 2015). Second, we will make the case that this approach is promising for WSD. Third, we will discuss the necessary modifications to the approach with respect to the choice of classifier, automatic label acquisition, and feature representation if it is to be used for WSD.

3.1 Ontological Constraints for Gloss Finding

To understand the gloss finding task, we have to define the terms "gloss" and "knowledge base". A gloss is a short natural language definition of a semantic category, e.g., "Apple is a fruit from the apple tree." A knowledge base is a directed acyclic graph in which each node belongs to one of three types: (1) categories such as "vegetable", "fruit", or "company"; (2) concrete instances of categories (so-called entities) such as "banana", "fruit:apple", and "company:apple"; (3) lexical strings, e.g., "apple", "Apple", and "Apple Inc." The different node types can only appear in certain places in the graph: the leaf nodes (and only the leaf nodes) are lexical strings, the parents of each leaf nodes are entities, and the ancestors of the entities are categories.

Note that the relationship between the entity and lexical string nodes is many-to-many whereas the the category and entity nodes form a n-ary tree structure with a unique root node (corresponding to the "Everything" category). In effect, this tree can be seen as an ontology where parent-child relationship between a category and an entity signifies an *is-a* relationship ("banana is-a fruit"), the parent-child relationship between categories signifies a *is-subcategory-of* relationship ("fruit is-subcategory-of food") and categories that are not in a ancestral relationship to each other can be seen as *is-mutually-exclusive-with* ("food is-mutually-exclusive-with organization"). The gloss finding task consists in assigning glosses from a large set to corresponding entities in the KB, e.g., "Apple is a fruit from the apple tree" should be assigned to "fruit:apple."

Dalvi et al. (2015) present GLOFIN, a hierarchical semi-supervised classification approach to the gloss finding task, and use to assign definitional sentences from Wikipedia to entities from the NELL KB. They assume that each gloss is identified by a head noun (e.g., "Apple is a fruit from the apple tree" is identified by "Apple")

which makes it possible to associate it with a corresponding lexical string in the KB. All the glosses that are associated with the lexical strings of a given entity are candidate glosses for that entity. If a gloss is candidate for only a single entity, it can be treated as unambiguous.

Using the unambiguous glosses as (noisy) labeled seeds and the remaining ambiguous glosses as unlabeled data, Dalvi et al. train a classifier using the expectation maximization technique. To take into account the ontological constraints between categories, they extend the E-step as follows: after computing the membership probabilities given the current estimates of the parameters (i.e., a soft assignment), they solve a mixed integer program (MIP) for each gloss. This MIP maximizes the probability score of the category membership of the gloss using indicator variables for the category membership as the optimization variables (i.e., a hard assignment) and treating the class membership probabilities and the ontological constraints as parameters. The MIP favors category membership assignments that are consistent with the ontology: if the gloss for "apple" is assigned to the "fruit:apple" entity, it should also be a member of the "fruit" and "food" category since these are the parent and grandparent of "fruit:apple", respectively. In the M-step, the parameters are re-estimated given the hard category assignments.

The results of Dalvi et al. show that GLOFIN works well for the gloss finding task and that using ontological constraints during the classification stage consistently provides an increase in performance compared to the flat version of the model. This is good news for us because the gloss finding task is quite similar to WSD once we replace the glosses with word contexts and the KB entities with WordNet senses. Hence, it is promising to try this approach, mutatis mutandis, on the WSD task.

The first component that has to be adapted is the KB. We will be using WordNet as the KB in our system.

3.2 WordNet

WordNet is a comprehensive lexical database for the English language (Miller, 1995). In WordNet, a word is a tuple of the form (string, sense). For instance, the string "goal" has four different senses ("goal#1", "goal#2", etc.) which means that there are four words with the string "goal". If there is only one sense associated with a given string, then the word with that string is called monosemous. On the other hand, if there are several senses associated with a given string, then the words with that string are called polysemous.

WordNet groups words that belong to the same part of speech (i.e., nouns, verbs, adjectives, and adverbs) and that share the same sense into sets. Each of these sets of synonyms (synsets) has a short natural language definition (gloss) and up to four example usages. For instance, the synset consisting of "goal#4", has the gloss "a successful attempt at scoring" and the example usage "the winning goal came with less than a minute left to play" (cf. Figure 2 in the appendix).

Noun synsets are interlinked via the semantic relationship of hypernymy (*is-a*

relationship, e.g., goal is a type of sports equipment). Hypernymy is transitive and thus induces a hierarchy onto the set of synsets. This hierarchy can be seen as an ontology and can be represented as a directed graph with synsets as vertices and hypernym relationships as directed edges.[1]

In our system, we will be using monosemous relatives of the target word to automatically acquire training examples (this will be described in depth in the next section). We will also extract a noun ontology from WordNet for each target lemma that will provide us with the hierarchical constraints that are used during the E-step of the classification.

3.3 Automatic Label Acquisition for WSD

One of the main difficulties of training a WSD system is the scarcity of high-quality labeled data. The existence of this so-called knowledge acquisition bottleneck is well-known fact and there have been efforts to find a way around by using techniques such as bootstrapping, active learning, and parallel corpora (see Agirre and Soroa (2009) for an overview).

Here, we focus our attention on the approach by Martinez et al. (2008) called automatic example acquisition that leverages the target word's monosemous relatives in WordNet to acquire (noisy) labeled examples. Although quite simple, this approach has been shown to be very effective in practice and it will be used in adapted form in our system.

The first step of the approach consists in collecting all the monosemous lemmas in WordNet. Then, each monosemous lemma is googled and the lines of text (so-called snippets) that appear underneath the first 1000 search results are collected into a corpus. At the end of the collection process, the corpus contains around 150 million examples.

Once the corpus has been constructed, it can be used to harvest labeled examples for specific target lemmas. This works as follows: for each sense of the target lemma, define a quota of examples and then collect the desired number of examples from the corpus by simple string matching. All the returned examples are ordered by their relationship to the target: synonyms are type 0, direct hyponyms are type 1, distant hyponyms have a type equal to their distance from the target, direct hypernyms have type 2, and siblings have type 3. To fill the quota, first all the examples of type 0 are selected, then all the examples from type 1 and so forth until the desired number of examples is met. The examples can then be used to train a supervised classifier.

One important question is how the quota for the different candidate senses should be set (i.e., what the sense prior should be). Martinez et al. try different settings: (1) uniform prior: equal number of examples per sense; (2) corpus prior: number

[1] Although the graph has a unique root (the "entity" synset), it is not a tree because some synsets have multiple hypernyms. Since these cases are rare, however, the graph can be treated as a tree for most practical purposes.

of examples per sense proportional to its frequency in the corpus; (3) automatic ranking, a more advanced technique described in McCarthy et al. (2004); (4) sense-tagged prior: number of examples proportional to its frequency in some hand-tagged corpus such as Semcor. They show that choosing the right prior has a large influence on the performance and that that priors that use supervision (i.e., sense-tagged priors) work better.

The underlying assumptions of this approach are: (1) the context of a target word should be similar to the context of its monosemous relatives; (2) the lower the type of the relative the stronger the similarity of the contexts (and thus the higher the reliability of the example). These assumptions seem to be justified given the good results the approach achieves on several WSD benchmarks.

3.4 Semi-Supervised Naive Bayes for Text Classification

The gloss finding algorithm that was discussed in Section 3.1 works with a number of different classifiers. For the WSD task, the most natural choice is the semi-supervised Naive Bayes model presented in Nigam et al. (2000) which was initially developed for semi-supervised text classification.

In this model, documents are represented as word-count vectors (i.e., as bags of words). The model assumes the following generative model for documents. For each document $\boldsymbol{x}_i$: (1) draw the document class c_j from a categorical distribution over M classes; (2) draw the document length $|\boldsymbol{x}_i|$ from a suitable distribution; (3) draw the word counts $\boldsymbol{x}_i$ from a class-specific multinomial distribution. The categorical distribution and the multinomial distributions are each given a symmetric Dirichlet prior.

Training this model consists in estimating the maximum a posteriori (MAP). If there is a class label for every document in the training set, then the posterior of the model parameters factorizes and the MAP can be obtained by simply taking smoothed ratios of empirical counts. By contrast, if the training set contains documents without a class label, there is coupling between the parameters which makes it impossible to factorize the posterior. In this case, Nigam et al. suggest the use of the EM technique. After building an initial classifier using only the labeled data, the algorithm alternates between the E-step, during which class membership probabilities for each unlabeled document are estimated given the current parameters, and the M-step, during which the parameters are re-estimated given the current membership probabilities. The algorithm terminates once the parameter estimates and class assignments are stable.

Nigam et al. note that the use of unlabeled data can help reduce variance in the estimation of parameters and thus increase classification accuracy in case there is only little labeled data available. However, there are also cases in which the use of unlabeled data leads to worse performance.

4 Data

We use the ukWaC corpus as the source of our unlabeled sentences. ukWaC is a general purpose corpus of English with over two billion tokens. It was created by crawling the UK top level domain (Ferraresi et al., 2008).

To be able to evaluate the performance of our system, we use a set of sense-annotated gold standard sentences for each of several target lemmas. This gold standard data has been made publicly available by Koeling et al. (2005). It contains about 300 examples sentences for each of 41 nouns. The examples come from three different sources: a balanced corpus (British National Corpus, BNC) and from two domain-specific corpora (FINANCE and SPORTS). The sentences were annotated by a group of two to four linguists who labeled the target word with either the gloss of the correct target synset or, in case they were not sure about the correct sense, as "unclear". The inter-annotator agreement on the complete data set is reported as being 65 percent (BNC: 60 percent, SPORTS: 65 percent, FINANCE: 69 percent).

5 Method

We will now present a semi-supervised model for WSD that is trained with automatically acquired examples. First, we describe how we extract ontologies from WordNet for each target word. Second, we show how we use these ontologies to automatically acquire labeled examples that can then be used as seeds in our semi-supervised model. Third, we show how we adapt the gloss finding algorithm to our problem setting by using constraints obtained from our ontologies. Finally, we discuss related topics such as the choice of feature representation, preprocessing, and the implementation of the model.

5.1 WordNet Ontologies

For each target word, we extract an ontology from WordNet according to a procedure that will be described in detail below. The ontology serves a dual purpose: (1) as a means to automatically acquire training examples; (2) to provide the constraints that are used during the classification stage itself. Thus, the ontology is crucial to our approach.

The extraction procedure can be described as follows. First, we add all the target synsets and their siblings. In a second step, we consider the target synsets' ancestors that lie on the shortest path between the target synset and the root. Of those synsets, we add the immediate parent of the target synset, the root and the three ancestors closest to the root (collectively called the toplevel) and discard the rest. As a result, each ontology contains the following synsets:

- The "entity" synset. It is the root node of the ontology.

- The target synsets and all their sister synsets. They are the leaf nodes of the ontology.
- The parent synset of each target synset that lies on the shortest path between the target synset and the root node. If there are several such paths, one of them is picked at random.
- The three ancestors of each parent synset that lie on the shortest path between the parent synset and the root node and that are closest to the root node (i.e., a child, grandchild and grand-grandchild of the root node). This assumes that the height of the parent synset is at least 4; if this is not the case, then as many ancestors as possible are added.

Figure 3 in the appendix shows a simplified version of the ontology for the target lemma "goal".

Note that we also include monosemous children for the purpose of example acquisition but not for classification (i.e., the classifier does not use any ontological constraints between monosemous children and other synsets).

The reason for including the target synsets' parent and sister synsets is to increase the number of monosemous lemmas that are closely related to the target synsets. The toplevel ancestors on the other hand are included because they represent general and distinct concepts such as "location", "event" and "unit" that are presumably helpful for disambiguation.

Early experiments have shown that including the midlevel ancestors (i.e., the ones that lie between the parent synsets and the toplevel) leads to deep ontologies (tens of levels) without improving performance. For this reason, we decided to not include them in the ontology. This has the welcome side effect that now all the target synsets lie on the same level (except in the rare cases in which the parent synset itself is part of the toplevel).

Because we only consider one path from each target synset to the root, our extraction method ensures that the resulting ontologies are trees (i.e., every synset has only a single hypernym). Forward edges are rare in WordNet anyway which means that enforcing a tree structure is an easy way to simplify the analysis of our WSD system without losing much information.

5.2 Acquisition of Examples

Our approach to acquiring labeled examples is similar in spirit to the approach of Martinez et al. (2008) that was described in Section 3.3. However, it differs from this approach with respect to the corpus that is used, the types of sentences we include, and the monosemous relatives we consider. Furthermore, since our model is semi-supervised, it can also benefit from using unlabeled data.

Corpus While Martinez et al. (2008) use web snippets to create their corpus, we use ukWaC, a large web corpus. Examples for a given head word are acquired by doing a string search for sentences that match the head word. Note that this means that example sentences always contain the monosemous relative which is not necessarily the case for web snippets.

Monosemous Relatives Monosemous lemmas are unambiguous by definition. Thus, if the head word of a sentence is a monosemous lemma, the sentence can always be assigned to the correct synset. This allows our system to generate training data from unlabeled sentences for all the synsets that contain at least one monosemous lemma.

We set a target of 300 sentences per synset and then search the ukWaC corpus for sentences that contain a monosemous lemma that belongs to the synset. The search stops as soon as the desired number of sentences is reached or, in case there are not sufficiently many sentences, when all of the ukWaC corpus has been searched.

We experimented with different numbers of examples and found that in general, setting a maximum number of examples per synset gives better results than setting a maximum number of examples per monosemous relative. The reason is that in the former scheme, synsets with several monosemous lemmas will not dominate synsets with only one or two monosemous relatives.

There are cases in which not enough examples from monosemous relatives could be found, either because a target had too few monosemous relatives or because the monosemous relatives were too rare in the corpus. In these cases, we substituted "first sense lemmas", i.e., lemmas of a synset that correspond to the first listed sense in WordNet. This is justified by the fact that the first listed sense of a lemma in WordNet is the most frequent sense in Semcor.

Column 6 of Table 1 shows the actual number of monosemous sentences for each target lemma generated this way.

Sentence Types In addition to sentences from ukWaC that contain the monosemous relative, we also include the gloss and examples usages for each synset in the ontology, regardless of whether the synset has a monosemous lemma or not. Since each synset only has a single gloss and only between zero and four example usages, however, the number of glosses and example usages is small compared to the number of monosemous sentences.

Unlabeled Sentences Our method is semi-supervised and can potentially benefit from unlabeled examples. For this reason, we added up to 300 sentences from ukWaC for each (polysemous) lemma in the ontology to the training set. Note that since the ontology only contains a small subset of all the WordNet synsets, there is no guarantee that the correct sense associated with the head word of an unlabeled

sentence is present in the ontology. Hence, there is a risk of injecting noise into the training process by adding unlabeled sentences.

5.3 Model

We use a semi-supervised Naive Bayes classifier as our underlying model. This classifier is similar to the one described in Section 3.4. The only differences are: (1) the classes are now given by the different senses that are present in the ontology; (2) the data consists of sentences, not documents; (3) the maximum number of EM iterations is fixed at three.

Using the technique presented in Dalvi et al. (2015) (cf. Section 3.1), we augment the E-step of this algorithm as follows. After calculating soft sense assignments (the usual E-step), we estimate a hard sense membership indicators y_{ij} for each sentence i and sense j. This indicator variables takes value 1 if sentence $\boldsymbol{x}_i$ has sense j and 0 otherwise. The membership vector of all the indicators for a given sentence i is given by $\boldsymbol{y}_i = [y_{i1}, \ldots, y_{iK}]$. Note that the membership vector is not a one-hot vector; any number of memberships can be activated at the same time.

The membership indicators are found using a mixed integer program (MIP) for each sentence. The program maximizes the probability score of the synset membership of the sentence using indicator variables for the synset membership as the optimization variables and treating the synset membership probabilities and the ontological constraints as parameters. The two types of constraints we use are *subset-of* and *mutex*. The *subset-of* relationship is between a synset and all of its ancestors whereas *mutex* is between a synset and any other synset that is neither an ancestor nor a descendant.

We experimented with a weighting schemes for the different sentence types. Under this scheme, higher quality data was assigned a higher weight when computing the empirical word counts, i.e., the empirical word counts of hand-labeled sentences were multiplied with a factor of 10, the ones of automatically labeled sentences with a factor of 3 and the ones of unlabeled sentences with a factor of 1. This did not have a large impact on the results, however, which is why we decided to not use a weighting scheme in the end.

Note that in this model, there no longer is a clear distinction between the training and prediction phase: the validation data is fed into the model at the same time as the training data and the classification is done at the same time.

5.4 Features

We use simple bag-of-words features: each sentence is represented by a term frequency (TF) vector of its constituent terms. There are two reasons for this: (1) NB is a generative model built on the assumption that the features are conditionally independent; notwithstanding the fact that this assumption is usually violated in practice, the problem of dependent features would be exacerbated by performing

more sophisticated feature extraction since these often introduce complex dependencies among features; (2) the focus of this paper is on evaluating the effectiveness of ontological constraints and not on investigating data representations.

5.5 Preprocessing

Before TF is calculated, sentences are sanitized as follows: (1) removal of non-ASCII characters; (2) converting to lowercase; (3) tokenization; (4) stopword removal; (5) lemmatization. The tokenizer used is the Treebank tokenizer since this is also what is used for sentences in the ukWaC. For sentences from other sources, we use the WordNet lemmatizer provided by the `stem.wordnet` module of Python's Natural Language Toolkit (`nltk`). For automatically labeled examples, we remove the target word from the sentence.

5.6 Implementation

For the supervised and semi-supervised NB model, we use the implementation in Python's `scikit-learn` library[2], version 0.17.1, by Pedregosa et al. (2011) and a modified version of the EM-extension[3] written by Mathieu Blondel, respectively. The model with ontological constraints is based on a modified version of the gloss finding code provided by Dalvi et al. (2015). This code uses the MIP solver provided in the MOSEK optimization toolbox[4] (version 7.0) to perform the optimization.

To query WordNet (version 3.0), we use the interface provided by Python's `nltk` library, version 3.2.1, by Bird et al. (2009). Pre- and postprocessing is done in Python using standard third-party libraries including `numpy`, `scipy`, `pandas`, `matplotlib`, `seaborn`, and `ipython`.

6 Experiment

6.1 Datasets

Training Table 1 contains basic statistics about the datasets for each of the twelve lemmas.

Validation Our validation data comes from the Koeling et al. (2005) dataset. We used the words *goal* and *fan* to develop our system and then added the following 10 randomly selected words to test performance: *chip, competition, conversion, division, fishing, level, manager, tie, top, transfer.* We only include sentences for which a majority of the experts agreed on the sense (i.e., at least 3 in the case of 4

[2] `scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html` (last visited on October 23, 2016).

[3] `gist.github.com/mblondel/f0789b921c98d0fe6868` (last visited on October 23, 2016).

[4] `www.mosek.com/products/mosek`

	#synsets		#lemmas		#sentences				
target	target	total	mono	total	mono	glosses	usages	gold	unlabeled
fan	3	175	149	282	20k	175	34	221	34k
goal	4	80	53	135	9k	80	45	296	23k
chip	9	74	73	146	10k	74	21	224	20k
competition	4	71	64	135	13k	71	34	278	20k
conversion	9	162	108	251	12k	162	63	261	39k
division	12	139	237	321	17k	139	33	233	24k
fishing	2	26	20	45	3k	26	7	206	7k
level	8	272	182	430	24k	272	135	202	67k
manager	2	15	23	35	4k	15	3	286	3k
tie	9	116	90	185	10k	116	28	231	25k
top	11	206	177	360	24k	206	99	179	50k
transfer	6	476	412	770	54k	476	163	236	91k

Table 1: Ontology and dataset statistics for all target lemmas

experts or at least 2 in the case of 2 or 3 experts) and for which the majority sense was not labeled "unclear". The number of validation sentences for each word can be seen in Table 1 (column "gold").

6.2 Baselines

We compare the performance of our model against two different kinds of baselines. The first kind of baseline are flat versions of the model that do not use the ontology during classification (i.e., fully-supervised and semi-supervised NB). Here, the idea is to isolate the effect of adding unlabeled data and of using ontological constraints at classification time. The second kind of baseline are state-of-the-art methods for knowledge-based WSD on the Koeling dataset. In this case, the idea is to compare the approach to unrelated but effective WSD methods.

Naive Bayes Our first baselines are the fully supervised and the semi-supervised version of the Naive Bayes model, called NB and NB-EM, respectively. NB only uses automatically labeled examples for the target lemma for training; NB-EM also includes unlabeled examples of the lemmas associated with the target synsets. The difference in performance between the two is an indicator of the usefulness of unlabeled data for the WSD task and the difference in performance between the semi-supervised NB model and our model shows the effectiveness of ontological constraints. Both NB and NB-EM use the corpus prior, i.e., the class distribution is estimated from the labeled training data using add-one smoothing. To get a better understanding of the effect of the prior, we also train a fully-supervised Naive Bayes classifier (NB-UNIF) that uses a uniform sense distribution.

Gloss Vectors Gloss vectors are a measure of semantic relatedness between WordNet synsets (Patwardhan and Pedersen, 2006). The vectors are constructed as second-order context vectors as follows: (1) for each content word in a augmented gloss, a first-order co-occurrence vector is created; (2) all the first-order co-occurrence vectors are summed up to form a second-order context vector. The augmented glosses themselves are constructed by concatenating the gloss of the target synset with the glosses of synsets that are closely related to the original synset by some relatedness measure such as the Leacock and Chodorow distance (Leacock and Chodorow, 1998) or the Extended Gloss Overlap (Banerjee and Pedersen, 2003).

We compare our results against the two gloss vector baselines that are reported in Pritsker (2014). The first baseline (GV) uses gloss vectors as features for a Naive Bayes classifier. The second baseline (GV-LCS) uses a modified version that only considers gloss vectors of the top-k most predictive features (so-called learned context selection model, see Pritsker et al. (2015)).

Personalized PageRank Personalized PageRank is a method to rank the nodes of a graph by their structural relevance. It can be applied to WSD as follows: (1) model the WordNet noun hierarchy as a mixed graph in which the synsets and lemmas are nodes, the semantic relationships between synsets are undirected edges, and there is a directed edge from each lemma to all its synsets; (2) distribute the initial probability mass equally over all the words in the context of the target; (3) run PageRank and select the target sense that has the highest score (Agirre and Soroa, 2009).

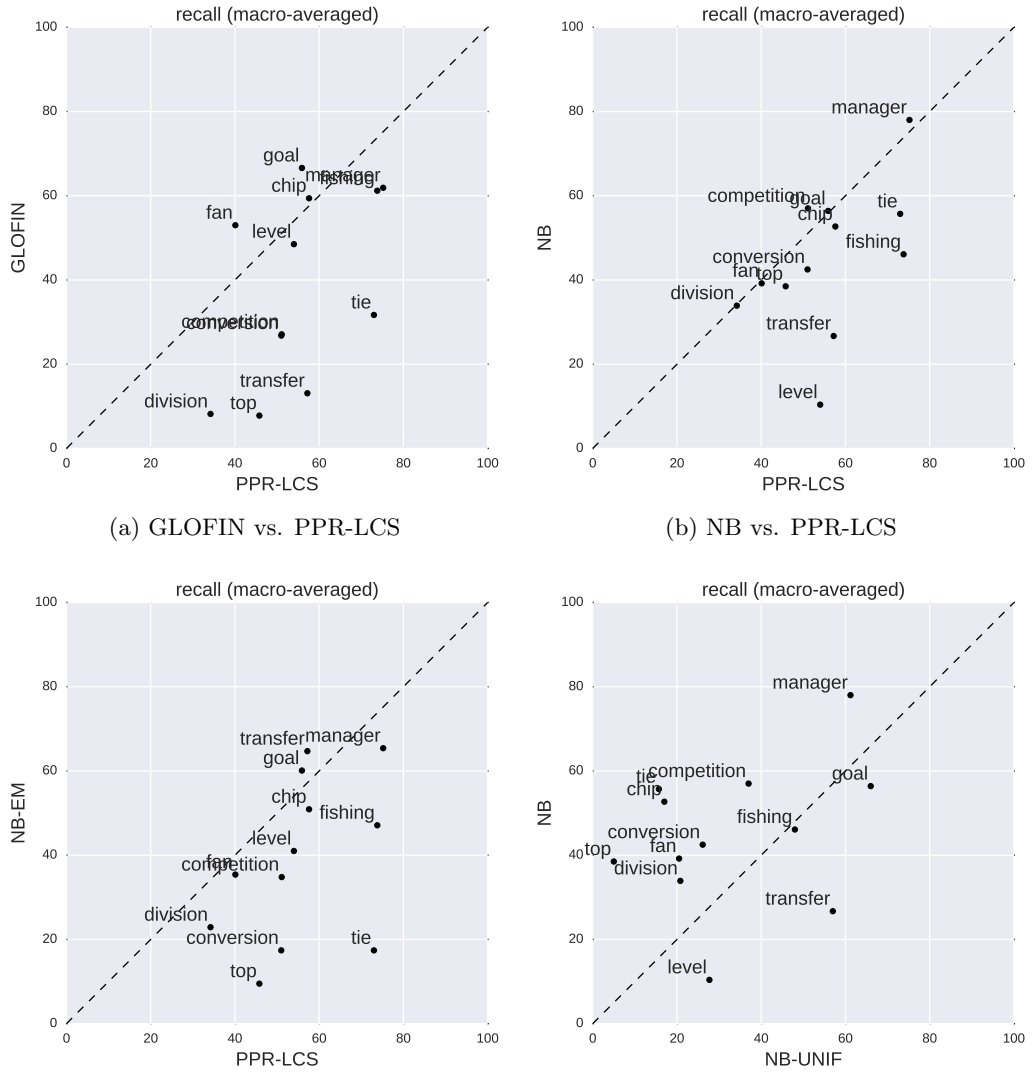
We compare our results against the two personalized PageRank baselines that are reported in Pritsker (2014): PPR, which is the standard version, and PPR-LCS which uses the learned context selection extension (Pritsker et al., 2015).

6.3 Evaluation Metric

We use macro-averaged recall as our evaluation metric. For a given class k, recall is defined as the number of correctly labeled sentences divided by the sum-total of sentences assigned to that class, i.e., $R_k = tp_k / (tp_k + fn_k)$, where tp and fn stand for true positive and false negative, respectively. To compute the macro-average, we simply compute the average recall over classes, i.e., $K^{-1} \sum_{k=1}^{K} R_k$.

7 Results

Figure 1 and Table 2 show the main results of the experiment.



(a) GLOFIN vs. PPR-LCS

(b) NB vs. PPR-LCS

(c) NB-EM vs. PPR-LCS

(d) NB vs. NB-UNIF

Figure 1: Performance comparisons for GLOFIN (a), NB (b), and NB-EM (c) against PPR-LCS baseline; for lemmas above the diagonal, the method outperforms the baseline, for lemmas below the diagonal, the method underperforms the baseline; (d) compares NB against NB-UNIF

	Recall (macro-averaged)							
lemma	GV	PPR	GV-LCS	PPR-LCS	NB	NB-UNIF	NB-EM	GLOFIN
fan	40.8	34.1	39.4	40.1	39.2	20.5	35.4	**53.0**
goal	53.7	54.6	54.7	55.9	56.4	66.0	60.1	**66.6**
chip	**63.8**	54.9	62.1	57.6	52.7	17.0	50.9	59.4
competition	43.2	47.1	40.6	51.1	**57.0**	37.0	34.8	27.1
conversion	33.7	**51.7**	39.8	51.0	42.5	26.1	17.4	26.8
division	34.9	**37.7**	25.9	34.2	33.9	20.8	22.9	8.2
fishing	46.6	72.3	47.8	**73.8**	46.1	48.0	47.1	61.2
level	45.5	50.5	42.6	**54.0**	10.4	27.7	41.0	48.5
manager	72.4	72.4	70.3	75.2	**78.0**	61.2	65.4	61.9
tie	23.5	66.1	27.0	**73.0**	55.7	15.7	17.4	31.7
top	33.0	**48.0**	32.4	45.8	38.5	5.0	9.5	7.8
transfer	31.4	47.9	36.0	57.2	26.7	57.0	**64.7**	13.1
average	43.5	53.1	43.2	**55.7**	44.8	33.5	38.9	38.8

Table 2: Main results (bold denotes best; *fan* and *goal* were used to develop and tune the GLFOIN model)

8 Discussion

As can be seen from Table 2, there is no uniformly dominant method: six different methods achieve the best result for at least one and at most three lemmas. Overall, PPR-LCS is the strongest method since it has the highest average recall. GLOFIN does well in the case of "fan" and "goal" (which is not surprising since these two lemmas were used to develop the system) but has very low performance for "division", "top", and "transfer."

Furthermore, we can observe a high high variance across methods. The variance across lemmas seems to be higher for NB, NB-UNIF, NB-EM, and GLOFIN, than it is for GV, PPR, GV-LCS, and PPR-LCS. In particular, the former all have lemmas with single-digit or near single-digit recall which is something that does not happen for the latter.

In the remainder of this section, we will try to analyze the effect of individual model components and modeling choices separately to gain a better understanding of overall performance.

Automatically Acquired Examples One possible explanation for the poor performance of GLOFIN could be that we use an unsuitable method to automatically acquire examples. If this were the case, we would expect all methods that use these

examples to perform badly. This, however, is not the case: NB, which is the simplest method that makes use of the examples, achieves a performance that is reasonably competitive for most lemmas and achieves the best performance for two lemmas (see Figure 1b).

A notable exception is the lemma "level" for which NB has the worst performance of all methods by far. A closer inspection reveals that automatic labeling fails in this case because the most prevalent sense of "level" in the validation data ("degree, grade, level") only has a single monosemous relative (a hyponym) which is "sun protection factor, SPF." Not surprisingly, this very specific monosemous relative fails to produce good training examples for the "degree, grade, level" sense of "level".

All in all, it seems unlikely that the poor performance of GLOFIN can be attributed to the way we automatically acquire examples.

Unlabeled Data A second reason why GLOFIN performs poorly could be the use of unlabeled data during the classification. As Nigam et al. (2000) have pointed out when they studied the effect of adding unlabeled examples for text classification, the results are highly domain-dependent. In particular, in cases where the generative model cannot capture important text properties, there is no clear correlation between classification accuracy and model likelihood and the addition of unlabeled data can actually hurt performance.

To isolate the effect of adding unlabeled data, we can study the performance of NB-EM which is the simplest version of our model that makes use of unlabeled data. What we get is a somewhat mixed picture: although NB-EM has a competitive performance for some lemmas (e.g., "goal" and "transfer"), it also has a clear below-average performance in a number of cases ("conversion", "tie", "top"; see Figure 1c). The bad performance could be due to the introduction of unrelated terms that are introduced when unlabeled sentences are added to the training process (so-called semantic drift, see Curran et al. (2007)).

Possible evidence for this hypothesis comes from looking at the words that achieve the highest probability score for a given class and the evolution of these words over iterations. For instance, the set of most predictive words for the "top side, upper side, upside" sense of "top" changes drastically after a complete iteration of EM when unlabeled sentences are added and the final classifier uses many seemingly non-discriminative terms such as "say", "good", "use", and "down." This might explain why the algorithm shows a strong and unwarranted bias towards this sense of "top".

Thus, it seems at least probable that the use of unlabeled sentences leads to worse performance in some cases.

Ontological Constraints Another reason for the bad performance of GLOFIN might be the use of ontological constraints during the classification. This would be somewhat surprising given that Dalvi et al. (2015) report that in their experiments,

ontological constraints always provided an increase in performance.

It is possible, however, that the WSD and the gloss finding task are less similar than they appear. While we do not have access to the detailed results of the gloss finding experiment, it is possible that the WordNet ontologies we use for WSD are unsuitable for this approach. In particular, there are lemmas for which some of the target synsets have a very large number of siblings. For instance, the "transferee" sense of "transfer" has 401 siblings whereas the other senses only have between 6 and 24 siblings each. The confusion matrix for "transfer" shows that GLOFIN indeed assigns 124 out of 236 (over 52 percent) of all examples to the sense "transferee" even though the true number of "transferee" examples is only 10 (less than 5 percent). This might be due to the fact that the parent of "transferee" which is "person, individual, someone, somebody mortal, soul" gets undue importance due to the number of its children which in turn encourages the model to assign many examples to the "transferee" sense. This is also supported by the fact that the confusion matrix of "transfer" for NB and NB-EM do not show the same bias for the "transferee" sense.

Sense Prior As Martinez et al. (2008) have shown, the sense prior is crucial to get good performance when using automatically acquired examples. It is natural to suspect that the prior also has a lot of influence on performance in our case. One experiment that shows the importance of the prior is to compare the results of NB which uses the corpus prior to the results of NB-UNIF which uses a uniform prior (see Figure 1d). As we can see, there are large differences in performance between the two methods for most of the lemmas; indeed, only in the case of "fishing" are the results remotely similar. The results of the corpus prior are good in the cases where the dominant sense is also the dominant sense in the Koeling dataset.

As discussed in Section 3.3, Martinez et al. show that in their experiments, using supervised priors obtained from hand-tagged corpora such as Semcor and Senseval-2 greatly increased the performance of their system. While it would be possible to use the same strategy for GLOFIN, it is doubtful whether this approach would yield better results. The reason is that the Koeling dataset is highly skewed and in all likelihood does not resemble either Semcor or Senseval-2 or any other tagged corpus for that matter. While we did not try to run GLOFIN with a Semcor prior, we nevertheless used the first sense in WordNet which corresponds to the most frequent sense in Semcor (see Martinez et al. (2008)) as a baseline (not reported) and only achieved subpar results for most lemmas.

Context Representation Finally, it might be the case that restricting our model to use simple bag-of-words features is hurting performance. As we discussed in Section 5.4, our decision to use a generative model discourages the use of more advanced features because they would introduce complex dependencies. Nevertheless, we experimented with both TF-IDF features (which are known to work well with NB in

under some circumstances) and gloss vector features. This, however, did not provide a better performance than simple bag-of-words features.

9 Conclusions and Future Work

We present a WSD system that combines elements from semi-supervised and knowledge-based WSD. In particular, our system uses automatically acquired examples and explicitly models ontological constraints during the classification stage. Unfortunately, our system is not able to outperform state-of-the-art knowledge-based baselines such as personalized PageRank.

While the complex interactions between different model components (automatically acquired examples, unlabeled data, ontological constraints, sense prior, context representation) make it difficult to pin down the exact reason for the disappointing performance, we can nevertheless study the effect of the individual components in particular cases. We hypothesize that that the most likely culprits are: (1) semantic drift stemming from the use of noisy unlabeled data; (2) highly unbalanced ontology structures; (3) different sense distributions in the training and validation set.

In order to test our hypothesis and hopefully increase the performance of our system, the next steps would be the following: (1) use a pruned version of the ontology that limits the number of siblings of a target synset, for instance by only considering siblings that are closely related to the target according to some similarity metric; (2) investigate better ways to choose the sense prior; (3) reduce semantic drift by only assigning unlabeled examples for which the model has reasonably high confidence.

10 Acknowledgements

I would like to thank my advisor William Cohen and my co-advisor Einat Minkov for their insight and expertise which greatly assisted this project. I am also very grateful to Bhavana Dalvi for letting me use her gloss finding code and for helping me to get started with the project.

References

Agirre, E. and Soroa, A. (2009). Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–41. Association for Computational Linguistics.

Banerjee, S. and Pedersen, T. (2003). Extended gloss overlaps as a measure of semantic relatedness. In *Ijcai*, volume 3, pages 805–810.

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python*. O'Reilly Media, Inc.

Curran, J. R., Murphy, T., and Scholz, B. (2007). Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, volume 3.

Dalvi, B., Minkov, E., Talukdar, P. P., and Cohen, W. W. (2015). Automatic gloss finding for a knowledge base using ontological constraints. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 369–378. ACM.

Ferraresi, A., Zanchetta, E., Baroni, M., and Bernardini, S. (2008). Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

Koeling, R., McCarthy, D., and Carroll, J. (2005). Domain-specific sense distributions and predominant sense acquisition. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 419–426. Association for Computational Linguistics.

Leacock, C. and Chodorow, M. (1998). Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283.

Martinez, D., De Lacalle, O. L., and Agirre, E. (2008). On the use of automatically acquired examples for all-nouns word sense disambiguation. *J. Artif. Intell. Res.(JAIR)*, 33:79–107.

McCarthy, D., Koeling, R., Weeds, J., and Carroll, J. (2004). Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 279. Association for Computational Linguistics.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134.

Patwardhan, S. and Pedersen, T. (2006). Using wordnet-based context vectors to estimate the semantic relatedness of concepts. In *Proceedings of the eacl 2006*

workshop making sense of sense-bringing computational linguistics and psycholinguistics together, volume 1501, pages 1–8. Trento.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Pritsker, E. W. (2014). *Learning Context Selection Models for Knowledge-based WSD.* Master's thesis (unpublished).

Pritsker, E. W., Cohen, W. W., and Minkov, E. (2015). Learning to identify the best contexts for knowledge-based wsd. In *EMNLP*, pages 1662–1667.

Appendix

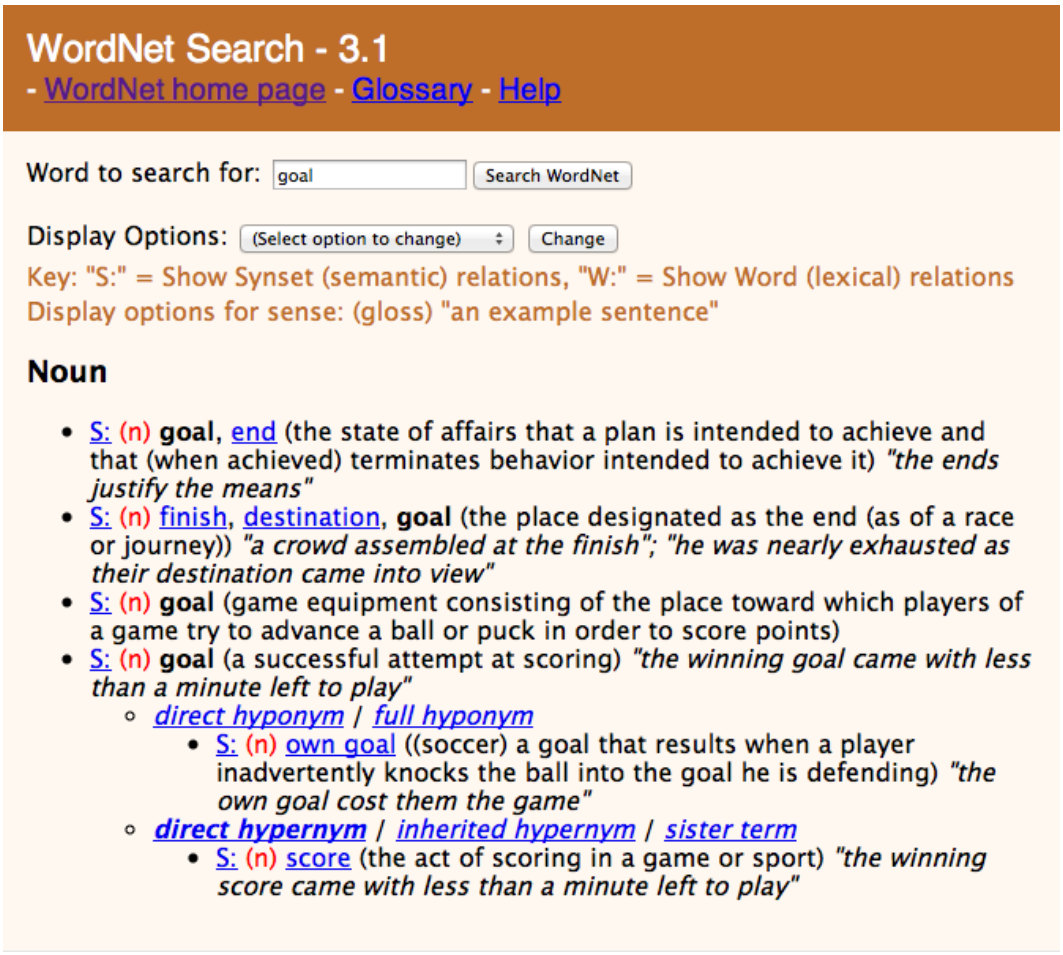


Figure 2: WordNet online search interface; shown is the search result for "goal": bullet points 1-4 are the four goal synsets; bullet point 5 is a direct hyponym of the last goal synsets; bullet point 6 is the direct hypernym of the last goal synset

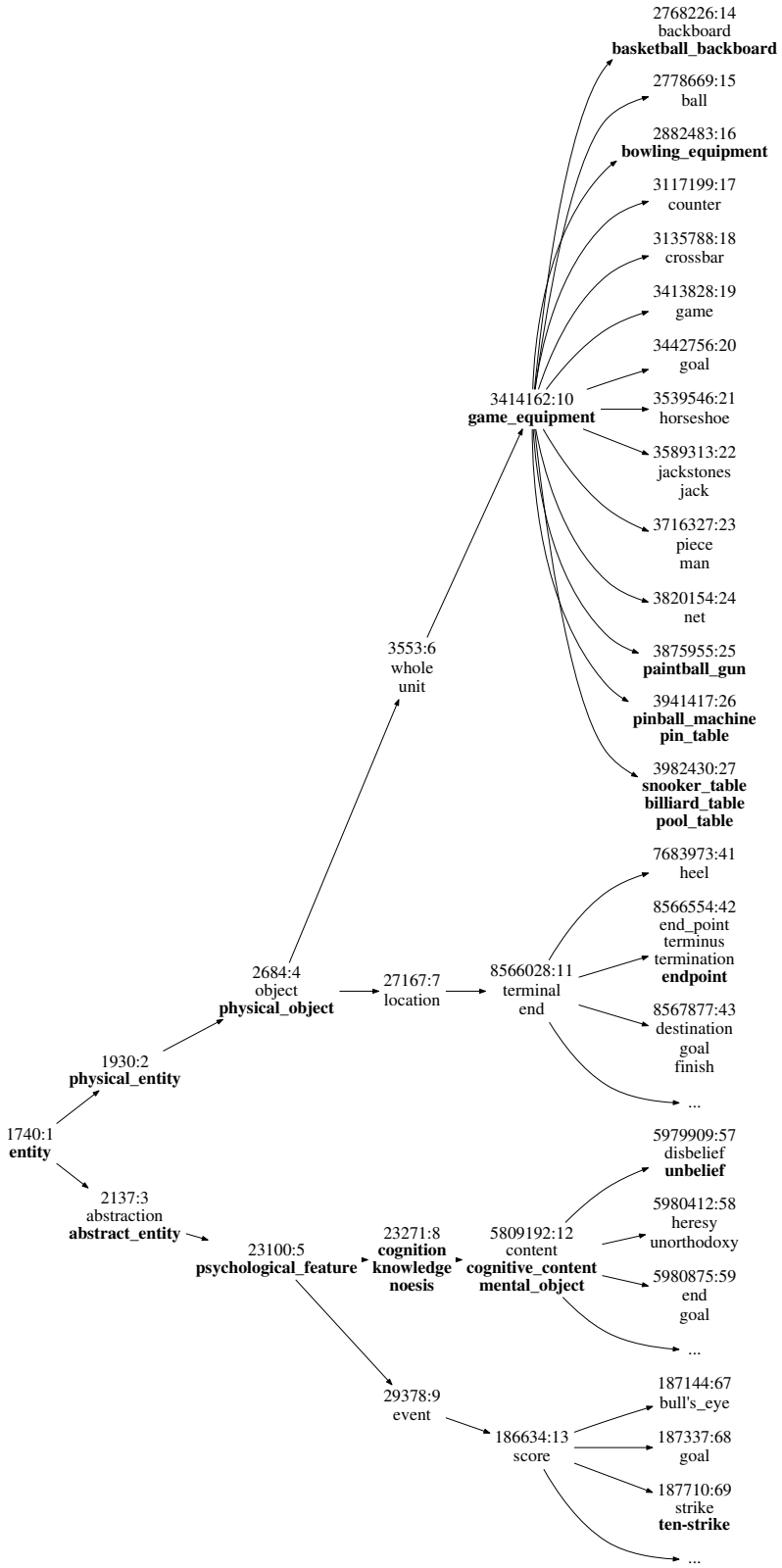


Figure 3: The ontology for target lemma "goal". Each synset is identified by a tuple of the form: *(wordnet_offset:class_label)*. Monosemous lemmas are printed in bold, Some of the target synsets' sister nodes have been removed for better legibility.