Word Sense Disambiguation with Neural Language Models.

Dayu Yuan Ryan Doherty Julian Richardson Colin Evans Eric Altendorf
Google Inc
Mountain View, CA
{dayuyuan,portalfire,jdcr,colinhevans,ealtendorf}@google.com

Abstract

Determining the intended sense of words in text – word sense disambiguation (WSD) – is a long-standing problem in natural language processing. In this paper, we present WSD algorithms which use neural network language models to achieve state-of-the-art precision. Each of these methods learns to disambiguate word senses using only a set of word senses, a few example sentences for each sense taken from a licensed lexicon, and a large unlabeled text corpus. We classify based on cosine similarity of vectors derived from the contexts in unlabeled query and labeled example sentences. We demonstrate state-of-the-art results when using the WordNet sense inventory, and significantly better than baseline performance using the New Oxford American Dictionary inventory. The best performance was achieved by combining an LSTM language model with graph label propagation.

1 Introduction

Word sense disambiguation (WSD) is a long-standing problem in natural language processing (NLP) with broad applications (Navigli, 2009). Supervised, unsupervised, and knowledge-based approaches have been studied for WSD (Navigli, 2009). However, for *all-words* WSD, where all words in a corpus need to be annotated with word senses, it has proven extremely challenging to beat the strong baseline, which always assigns the most frequent sense of a word without considering the context (Pradhan et al., 2007; Navigli et al., 2007; Navigli, 2009; Navigli et al., 2013; Moro and Navigli, 2015). Given the good performance of published supervised WSD systems when provided with significant training data on specific words (Navigli, 2009; Zhong and Ng, 2010), it appears lack of sufficient labeled training data for large vocabularies is the central problem.

To address the difficulty of obtaining large amounts of labeled data, in this work we study semi-supervised approaches. We use vector representations of words and contexts created using a neural-network based language model (NNLM) trained over a large unlabeled text corpus. We define a *context* to be a sentence minus a held out word. The NNLM produces a *context vector* from each context. We then classify based on the cosine similarity of context vectors.

We study two NNLMs in this paper. The first model is the continuous bag of words model (CBOW) (Mikolov et al., 2013). We construct the context vector as a weighted sum of the CBOW word vectors for the words in the context. This model is simple and efficient, but does not account for word order. The second model is based on a Long Short Term Memory model (LSTM) (Hochreiter and Schmidhuber, 1997). We train the LSTM on a large unlabeled corpus of text to predict a held-out word given the context (as a sequence). We use a hidden layer in the network, during prediction, as a vector representation for the context.

We also present an algorithm for semisupervised learning, using label propagation (Talukdar and Crammer, 2009; Ravi and Diao, 2016) to label unlabeled sentences based on their similarity to labeled ones. This allows us to better estimate the distribution of word senses, obtaining more accurate decision boundaries and higher classification accuracy.

Our experiments show that using the LSTM language model achieves significantly higher precision than the CBOW language model, especially on verbs and adverbs. This suggests that sequential order information is important to discriminating senses of verbs and adverbs. The best performance was achieved by using an LSTM language model with label propagation. Our algorithm outperforms the baseline by more than 10% (0.87 vs. 0.75).

Organization: We review related work in Section 2. We introduce our supervised WSD algorithms in Section 4, and the semi-supervised WSD algorithm in Section 5. Experimental results are discussed in Section 6. We provide further discussion and future work in Section 7.

2 Related Work

The recent development of large lexical resources, such as WordNet (Fellbaum, 1998) and BabelNet (Navigli and Ponzetto, 2012), has enabled knowledge-based algorithms which show promising results on all-word prediction tasks (Ponzetto and Navigli, 2010; Navigli et al., 2013; Moro and Navigli, 2015). WSD algorithms based on supervised learning are generally believed to perform better than knowledge-based WSD algorithms, but they need large training sets to perform well (Pradhan et al., 2007; Navigli et al., 2007; Navigli, 2009; Zhong and Ng, 2010). Acquiring large training sets is costly. In this paper, we show that a supervised WSD algorithm can perform well with a small number (e.g., 20) of training example sentences per sense.

In the past few years, much progress has been made on using neural networks to learn word embeddings (Mikolov et al., 2013; Levy and Goldberg, 2014), to construct language models (Mikolov et al., 2011), perform sentiment analysis (Socher et al., 2013), machine translation (Sutskever et al., 2014) and many other NLP applications. For the WSD problem, Rothe and Schütze (2015) extended word embeddings to word sense or synset embeddings, and trained a SVM classifier (Zhong and Ng, 2010) with word sense and synset embeddings as features. Taghipour et al. (2015) trained a feedforward neural network with automatically labeled training samples to learn word embeddings for WSD. However, the quality of the automatically labeled data is hard to control.

In this work, we take a different approach. Our neural networks are trained with large corpora of *unlabeled* data to predict a word from its surrounding context. The context vector, produced by the NNLM, captures the syntactic and semantic patterns of the context. Using a nonparametric nearest neighbor classifier with those vectors, our WSD classifier achieves high performance with little training data by taking advantage of the high-level features learned from neural-network based language models.

3 Classifying Using Example Sentences

The methods which we propose require one or more example sentences for each sense in the inventory to which we wish to classify. Our supervised WSD algorithms (Section 4) classify a word by finding the example sentences which are most similar to the sentence in which the word appears (Figure 3a). To overcome relative sparseness in the space of example sentences, we also present a semisupervised method (Section 5), which augments the example sentences with a large number of unlabeled sentences from the web. Sense labels are then propagated from the example sentences to the unlabeled sentences (Figure 3).

4 Supervised WSD

4.1 Word Vector Similarity

Distributional methods, in particular using word vectors trained using neural networks, have been shown to be extremely effective in NLP tasks (Turian et al., 2010; Baroni et al., 2014). The CBOW model is trained to predict a target word with its context as input (Mikolov et al., 2013). CBOW first computes a context vector by adding the vectors of the context words, and then predicts the target with the context vector (Figure 1). We use the context vector from CBOW for WSD.

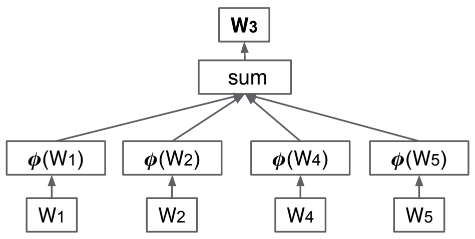


Figure 1: CBOW: $\phi(w)$ is the embedding for word w_i.

We used 1000-dimensional embeddings from a 100-billion-word news corpus, trained using word2vec (Mikolov et al., 2013). The vocabulary consists of the most frequent $1,000,000$ words, without lemmatization or case normalization. To obtain a context vector, we compute the sum of the embedding vectors for the words in the context, weighted by the log of their reciprocal rank in the corpus. This is similar to (Gabrilovich and Markovitch, 2007), which computes an embedding for a context by weighting the constituent word embeddings using their TF-IDF in some corpus – the purpose is to down-weight frequent words.

To compute a vector for a given sense, we average the context vectors for the example sentences of that sense. To classify, we find the sense vector with maximal cosine similarity to the context vector of the target word. For example, the query in Table 1 should be annotated as 'sense#1' because of its similarity to the first sense, 0.44, is higher than the similarity to the other two senses, 0.31 and 0.21. (We also tried classifying to the sense which contains the single example sentence with maximal cosine similarity, but this performed no better.)

id	sentence	label
1	Employee compensation is offered in the form of cash and/or *stock*.	sense#1
2	The *stock* would be redeemed in five years, subject to terms of the company's debt.	
3	These stores sell excess *stock* or factory overruns .	sense#2
4	Our soups are cooked with vegan *stock* and seasonal vegetables.	sense#3
query	In addition, they will receive *stock* in the reorganized company, which will be named Ranger Industries Inc.	?

Table 1: Four examples and one query for the word "stock".

More formally, let $\phi(w)$ denote the vector for a word w obtained from the trained embeddings (or 0 if w is not in the vocabulary), and $r(w)$ denote the rank of w in the training corpus, sorted by frequency. Then the context $\overline{w} = w_1, ..., w_{k-1}, w_k, w_{k+1}, ..., w_n$, where w_k is the word we wish to classify, has the context vector

$$\phi(\bar{(w)}) = \sum_{i \neq k} log(1 + r(w_i)) \times \phi(w_i).$$

For a word w with part of speech p, let $s_i^{w,p}$ be the word senses which have the same lemma and part of speech as w, and $s_{ij}^{w,p}$ be the example sentences associated with $s_i^{w,p}$. If $cos(.,.)$ denotes the cosine similarity function, then the classification of a word, w, appearing with part of speech p in a context $\overline{w}$ is $s_i^{w,p}$, where

$$i = \arg\max cos(\phi(\overline{w}), \sum_j \phi(s_{ij}^{wp})) \qquad (1)$$

The evaluation results are shown in Table 5. This method is limited by the fact that it sees only a bag of context words. Next we investigate a stronger language model which considers word order.

4.2 LSTM Language Model

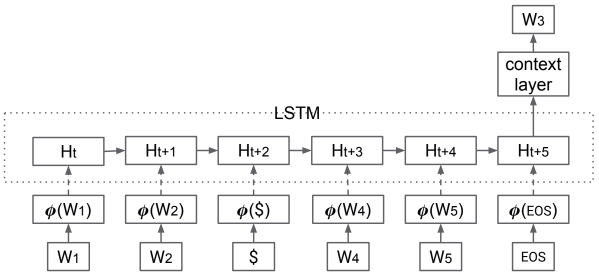


Figure 2: LSTM: Replace the focus word w_3 with a special symbol \$ and predict w_3 at the end of the sentence. Both the embedding and the context layer have p nodes. The dimension of layer H is h.

Neural networks with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) make good language models which do consider word order (Sundermeyer et al., 2012). We train an LSTM language model to predict the held-out word in a sentence (Figure 2). The LSTM was trained on a news corpus of about 100 billion tokens, with a vocabulary of almost $1,000,000$ words. Words in the vocabulary are neither lemmatized nor case normalized.

The behavior of the LSTM can be intuited by its predictions. Table 2 shows the top 10 words predicted by an LSTM language model for the word 'stock' in example and query sentences from Table 1.

id	top 10 predictions from LSTM	label
1	cash, stock, equity, shares, loans, bonus, benefits, awards, equivalents, deposits	sense#1
2	bonds, debt, notes, shares, stock, balance, securities, rest, Notes, debentures	
3	inventory, goods, parts, sales, inventories, capacity, products, oil, items, fuel	sense#2
4	foods, food, vegetables, meats, recipes, cheese, meat, chicken, pasta, milk	sense#3
query	shares, positions, equity, jobs, awards, representation, stock, investments, roles, funds	

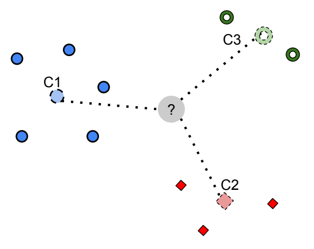
Table 2: Top predictions of 'stock' in sentences from Table 1

In our initial experiments, we computed similarity between two contexts by the overlap between their bags of predicted words. For example, the top predictions for the query overlap most with the LSTM predictions for 'sense#1', so we predict that 'sense#1' is the correct sense.

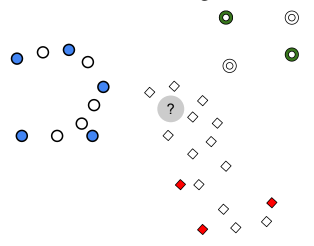
This bag of predictions, while easily interpretable, is just a discrete approximation to the internal state of the LSTM when predicting the held out word. We therefore directly use the LSTM's hidden layer from which the bag of predictions was computed as a representation of the context (see Figure 2). Given context vectors extracted from the LSTM, classification is performed by finding the sense with the most similar example sentences as in Equation 1.

As can be seen in Tables 5 and 6, this model has significantly better performance than the CBOW model described in Section 4.1.

5 Semi-supervised WSD



(a) nearest neighbor



(b) semi-supervised WSD with label propagation

Figure 3: Nearest neighbor classifier (top) vs. semi-supervised classifier (bottom): Filled nodes represent labeled sentences. Unfilled nodes represent unlabeled sentences.

The non-parametric nearest neighbor algorithm which we described in Section 4 has the following drawbacks:

- It assumes a spherical shape for each sense cluster, being unable to accurately model the decision boundaries given the limited number of examples.
- It has no training data for, and does not model, the sense prior, omitting an extremely powerful potential signal.

We next study a semi-supervised algorithm and show that by selecting and labeling sentences randomly sampled from the web, we can better estimate the shape and size of the sense clusters.

A *label-propagation graph* consists of (a) vertices with a number of labeled seed nodes and (b) undirected weighted edges. Label propagation (LP) (Talukdar and Crammer, 2009) iteratively computes a distribution of labels on the graph's vertices to minimize a weighted combination of:

- The discrepancy between seed labels and and their computed labels distributions.
- The disagreement between the label distributions of connected vertices.
- A regularization term which penalizes distributions which differ from the prior (by default, a uniform distribution).

We construct a graph for each lemma where each vertex represents a sentence in which the lemma occurs. We first create and label vertices for labeled example sentences. In addition, we sample 1000 unlabeled sentences containing the lemma from a large text corpus. Vertices for sufficiently similar sentences (based on criteria discussed below) are connected by an edge whose weight is the cosine similarity between the respective context vectors, using either the CBOW or LSTM language model.

To classify an occurrence of the lemma, we create an additional vertex for the new sentence and run LP to propagate the sense labels from the seed vertices to the unlabeled vertices.

Figure 3 (b) illustrates the graph configuration. Spatial proximity represents similarity of the sentences attached to each vertex and the shape of each node represents the word sense. Filled nodes represent seed nodes with known word senses. Unfilled nodes represent sentences with no word sense label, and the ? represents the word we want to classify.

With too many edges, sense labels propagate too far, giving low precision. With too few, sense labels do not propagate sufficiently, giving low recall. We found that the graph has about the right density for common senses when we ranked vertex pairs by similarity and connected the top 5%. This may still leave rare senses sparsely connected, so we additionally added edges to ensure that every vertex is connected to at least 10 other vertices.

This WSD algorithm produced the best precision and recall (Table 5, 6). Since it requires running LP for every classification, the algorithm is slow compared to the algorithms we previously described.

6 Experiments

We evaluated performance with several different experiments. We compared the CBOW and LSTM NNLMs with and without LP. We trained on several different corpora using two different inventories. We also compare the performance of our algorithms to a selection of baseline WSD algorithms and to SemEval 2013 and 2015 results.

6.1 Word Sense Inventory

In this work we use the New Oxford American Dictionary (NOAD) (Stevenson and Lindberg, 2010). The NOAD focuses on American English and is based on the Oxford Dictionary of English (ODE) (Stevenson, 2010). It distinguishes between coarse (*core*) and fine-grained (*sub*) word senses in the same manner as ODE. Previous investigations (Navigli, 2006; Navigli et al., 2007) using the ODE have shown that coarse-grained word senses induced by the ODE inventory address problems with WordNet's fine-grained inventory, and that the inventory is useful for word sense disambiguation.

For our experiments, we use NOAD's core senses, and we also use lexicographer-curated example sentences from the Semantic English Language Database (SELD)[1], provided by Oxford University Press. Table 3 shows the total number of polysemes (more than one core sense), average number of senses per polyseme and average number of example sentences per sense in NOAD/SELD (hereafter, "NOAD"). We manually annotated all words of the English language SemCor (Miller et al., 1993) corpus and MASC [2] corpora with NOAD word senses in order to evaluate performance.

	noun	verb	adj.	adv.
polyseme count	8097	2124	2126	266
sense count/polyseme	2.46	2.58	2.30	2.47
example count/sense	15.67	27.38	19.40	20.30

Table 3: Polyseme in NOAD

We also evaluate our techniques using WordNet as the inventory. To overcome the general lack of example sentences in WordNet, we developed a partially crowd-sourced mapping from NOAD to WordNet. This allows us to produce WordNet sense labels for NOAD example sentences, and therefore use those example sentences to classify directly into WordNet space.

[1]http://oxfordgls.com/our-content/english-language-content/

[2]http://www.anc.org/MASC/About.html/

6.2 NOAD Experiment

Data: We use NOAD example sentences as labeled training data. We evaluate on SemCor and MASC. We evaluate all polysemous words in the evaluation corpus. Table 4 lists the number of polysemous words and average number of candidate senses per word in NOAD, SemCor and MASC.

	example count (in 1000's)					sense count
dataset	all	n.	v.	adj.	adv	
NOAD	580	312	150	95	13	3.1
SemCor	115	38	57	11.6	8.6	4.1
MASC	133	50	57	12.7	13.6	4.2

Table 4: Number of examples in each dataset and the average sense count per example.

We compare our algorithms with five baseline algorithms:

- First sense: Label word w with w's first NOAD sense, which typically is the most popular sense.
- Most frequent sense: Compute the sense frequency (from a labeled corpus) and label word w with w's most frequent sense.
- Gloss overlap: Annotate a word in a given context with the sense whose definition overlaps the most with the context.
- Fuzzy Gloss overlap: Similar to Gloss overlap, but measure the overlap of two words with the cosine similarity between two words' CBOW word vectors.
- Lesk: Annotate a word in a given context with the sense whose gloss shares the largest number of common words with the glosses of the other words in the context (Lesk, 1986).

We train the CBOW and LSTM language models from a 100 billion word news corpus. The CBOW word vectors are of dimension 1024. The LSTM model has 2048 hidden units, and inputs are 512-dimensional word vectors. The LSTM learning rate is 0.1. We experimented with other learning rates, and observed no significant performance difference under different learning rates after the training converges.

eval data		SemCor					MASC				
model	train data	all	n.	v.	adj.	adv.	all	n.	v.	adj.	adv.
Lesk Score		0.634	0.709	0.564	0.705	0.666	0.628	0.718	0.539	0.703	0.594
Gloss Overlap		0.603	0.651	0.551	0.652	0.662	0.583	0.667	0.489	0.662	0.589
Fuzzy Gloss		0.562	0.653	0.478	0.652	0.584	0.552	0.663	0.427	0.652	0.567
First Sense		0.635	0.709	0.564	0.714	0.666	0.628	0.715	0.539	0.712	0.594
Frequent Sense	SemCor						0.753	0.799	0.713	0.758	0.741
Frequent Sense	MASC	0.752	0.751	0.749	0.737	0.789					
CBOW	NOAD	0.709	0.783	0.657	0.736	0.693	0.671	0.790	0.562	0.724	0.638
CBOW	SemCor						0.692	0.806	0.592	0.754	0.635
CBOW	NOAD,SemCor						0.678	0.808	0.565	0.753	0.604
CBOW	MASC	0.698	0.785	0.619	0.766	0.744					
CBOW	NOAD,MASC	0.695	0.801	0.605	0.767	0.719					
LSTM	NOAD	0.786	0.796	0.782	0.781	0.784	0.786	0.805	0.772	0.776	0.786
LSTM	SemCor						0.799	0.843	0.767	0.808	0.767
LSTM	NOAD,SemCor						0.812	0.846	0.786	0.816	0.798
LSTM	MASC	0.810	0.825	0.799	0.809	0.825					
LSTM	NOAD,MASC	0.821	0.834	0.814	0.818	0.821					
CBOW+LP	NOAD	0.642	0.733	0.554	0.705	0.725	0.643	0.752	0.524	0.726	0.664
CBOW+LP	NOAD,SemCor						0.780	0.830	0.740	0.803	0.749
CBOW+LP	NOAD,MASC	0.783	0.792	0.774	0.782	0.802					
LSTM+LP	NOAD	0.822	0.859	0.800	0.817	0.816	0.831	0.865	0.806	0.825	0.821
LSTM+LP	NOAD,SemCor						**0.872**	**0.897**	**0.852**	**0.865**	**0.868**
LSTM+LP	NOAD,MASC	**0.873**	**0.883**	**0.870**	**0.858**	**0.874**					

Table 5: F1 scores of our algorithms in comparison with baselines

Table 5 compares the F1 scores of the CBOW, LSTM and baseline algorithms. Both CBOW and LSTM beat the baselines by a wide margin. LSTM outperforms CBOW by more than 10% over all words, where most of the gains are from verbs and adverbs. The results suggest that syntactic information, which is well modeled by LSTM but ignored by CBOW, is key to distinguishing word senses of verbs and adverbs.

We also compute F1 score per sense and average the F1 scores across all word senses to compute **macro** F1 scores. Table 6 shows the macro scores. Both CBOW and LSTM beat the baseline algorithms. Also, LSTM outperforms CBOW by a wide margin. The first sense classifier and the most frequent sense classifier perform poorly on the macro scores since they never predict the less frequent senses.

6.2.1 Training data

By default, the WSD classifier uses the NOAD example sentences as training data. We build a larger training dataset by adding labeled sentences from SemCor and MASC, and study the change of F1 scores in Table 5 and Table 6. Across most part of speech tags and datasets, both the micro and macro F1 scores increase after adding more training data. We further test our algorithm by using SemCor (or MASC) as training data (without NOAD examples). The SemCor (or MASC) trained classifier is on a par with the NOAD trained classifier on F1 score. However, the macro F1 score of the former is much lower than the latter, because of the limited coverage of rare senses and words in SemCor and MASC.

6.2.2 Change of language model capacity

In this experiment, we change the LSTM model capacity by varying the number of hidden units h and the dimensions of the input embeddings p and measuring F1. Figure 4 shows strong correlation between F1 and the capacity of the language model. However, larger models are slower to train and use more memory. To balance the accuracy and resource usage, we use the second best LSTM model ($h = 2048$ and $p = 512$) by default.

6.3 Semi-supervised WSD

We evaluate our semi-supervised wsd classifier in this subsection.

For each word, we collect three types of sentence:(1) seed sentences: labeled sentences from the training datasets, (2) 1000 unlabeled sentences randomly sampled from the web, and (3) evaluation sentences with hidden gold sense labels.

We construct the graph as described in Section 5 and run LP to propagate sense labels from the seed vertices to the unlabeled vertices. We eval-

eval data		SemCor					MASC				
model	**train data**	**all**	**n.**	**v.**	**adj.**	**adv.**	**all**	**n.**	**v.**	**adj.**	**adv.**
Lesk Score		0.506	0.535	0.456	0.530	0.410	0.509	0.535	0.469	0.536	0.371
Gloss Overlap		0.538	0.568	0.488	0.561	0.436	0.540	0.566	0.497	0.570	0.389
Fuzzy Gloss		0.581	0.614	0.547	0.576	0.449	0.583	0.618	0.542	0.574	0.474
First Sense		0.506	0.535	0.456	0.529	0.410	0.508	0.534	0.466	0.534	0.368
Frequent Sense	SemCor						0.526	0.550	0.495	0.534	0.419
Frequent Sense	MASC	0.512	0.542	0.468	0.521	0.424					
CBOW	NOAD	0.718	0.751	0.695	0.707	0.540	0.728	0.757	0.700	0.730	0.582
CBOW	SemCor						0.620	0.629	0.629	0.610	0.514
CBOW	NOAD,SemCor						0.736	0.762	0.710	0.743	0.593
CBOW	MASC	0.590	0.608	0.581	0.582	0.486					
CBOW	NOAD,MASC	0.724	0.756	0.700	0.713	0.556					
LSTM	NOAD	0.769	0.791	0.759	0.751	0.672	0.780	0.791	0.768	0.780	0.726
LSTM	SemCor						0.656	0.663	0.668	0.643	0.581
LSTM	NOAD,SemCor						**0.796**	0.805	**0.790**	**0.794**	**0.742**
LSTM	MASC	0.631	0.653	0.606	0.617	0.600					
LSTM	NOAD,MASC	**0.782**	**0.803**	**0.774**	**0.761**	**0.688**					
CBOW+LP	NOAD	0.627	0.654	0.600	0.631	0.475	0.638	0.666	0.600	0.653	0.492
CBOW+LP	NOAD,SemCor						0.649	0.675	0.618	0.664	0.496
CBOW+LP	NOAD,MASC	0.635	0.660	0.611	0.641	0.490					
LSTM+LP	NOAD	0.767	0.796	0.749	0.744	0.645	0.780	0.803	0.759	0.770	0.690
LSTM+LP	NOAD,SemCor						0.790	**0.813**	0.771	0.775	0.709
LSTM+LP	NOAD,MASC	0.774	0.801	0.757	0.752	0.670					

Table 6: Macro F1 scores of our algorithms in comparison with baselines

uate the performance of the algorithm by comparing the predicted labels and the gold labels on eval nodes. As can be observed from Table 5, LP did not yield clear benefits when using the CBOW language model. We hypothesize that this is because LP is sensitive to the quality of the graph distance metric.

We did see significant improvements using LP with the LSTM language model. As can be seen in Table 5, LP substantially improves classifier F1 when the training datasets are SemCor+NOAD or MASC+NOAD. Running LP with the LSTM language model achieves the highest F1 scores.

As discussed in Section 5, the system benefits from (1) explicitly modeling the sense prior and (2) using the unlabeled nodes to better model sense distributions and decision boundaries. The distribution of example sentences in NOAD does not give a good approximation to the sense prior in free text, so LP would not give much benefit without the unlabeled web data.

There is a trade-off between the macro F1 (an average of the per-sense F1) and micro F1 (computed per example, with more frequent senses carrying more weight). Table 6 shows the macro F1 scores of the LP classifiers. LP generally results in lower macro performance, although the decrease is not severe.

6.4 SemEval Tasks

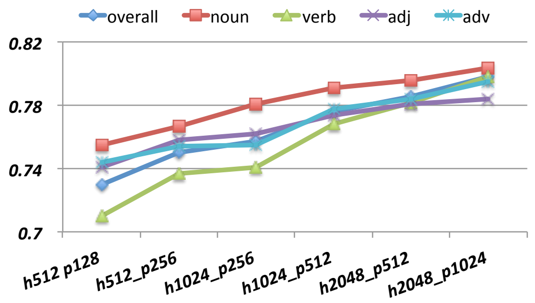
In this section, we study the performance of our classifiers on SemEval-2013 Task 12 (Navigli et al., 2013) and SemEval-2015 task 13 (Moro and Navigli, 2015). These two tasks use WordNet as the sense inventory for English WSD. We use a mapping to label NOAD example sentences with WordNet senses (see Section 6.1) for training. We also use SemCor (annotated with WordNet senses) for training. For a fair comparison with related works, the classifiers are evaluated on all words (both polysemous and monosemous).

Table 7 shows the results of Sem-Eval 2013. Our proposed algorithms outperform UMCC-DLSI, the best WSD algorithm reported in Sem-Eval 2013. The LP classifier with an LSTM language model has the highest score.

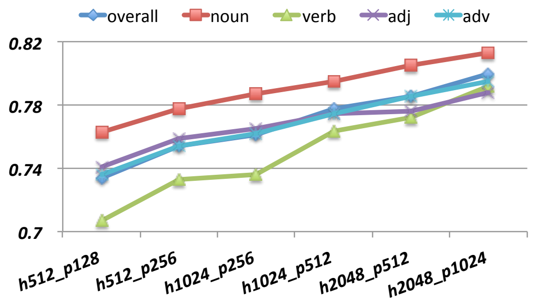
algorithm	precision	recall	F1
Most Frequent Sense	0.630	0.630	0.630
UMCC-DLSI	0.649	0.645	0.647
CBOW	0.661	0.661	0.661
CBOW+LP	0.676	0.676	0.676
LSTM	0.660	0.660	0.660
LSTM+LP	**0.692**	**0.692**	**0.692**

Table 7: Evaluate SemEval-2013 English Dataset. Most Frequent Sense baseline uses the WordNet most frequent sense.

Table 8 shows the results of Sem-Eval 2015. The LP classifier with an LSTM language model



(a) eval on SemCor



(b) eval on MASC

Figure 4: Micro F1 scores of LSTM models with different capacity: h is the number of hidden units; p is the embedding dimension.

achieves the highest scores on nouns and verbs as well as overall F1.

algorithm	all	n.	v.	adj.	adv.
LIMSI	0.647				0.795
DFKI		0.703	0.577		
UNIBA				0.790	
BFS Baseline	0.663	0.667	0.551	**0.821**	0.825
CBOW	0.684	0.678	0.618	0.765	0.766
CBOW+LP	0.685	0.678	0.587	0.789	**0.826**
LSTM	0.695	0.702	0.587	0.795	0.778
LSTM+LP	**0.718**	**0.717**	**0.634**	0.819	0.778

Table 8: F1 Scores of SemEval-2015 English Dataset. The BFS baseline uses BabelNet first sense.

7 Discussion and Future Work

Our approach to WSD does not rely on large labeled data sets, or on structured semantic resources, both of which are costly to curate. Instead, it leans on powerful language models learned from large unlabeled corpora, and on label propagation over unlabeled sentences using representations learned by the language model. This enables us to exploit small (e.g. 20 examples per sense) labeled data sets which may be licensed from existing lexicographic sources.

Several unanswered questions suggest lines of future work. We have not yet studied the relationship between number of labeled training examples and performance. Since our general approach is amenable to incorporating any language model, further developments in NNLMs may permit increased performance. We would also like to better understand the *limitations* of language modeling for this task: we expect there are situations – e.g., in idiomatic phrases – where per-word predictions carry little information.

We believe our model should generalize to languages other than English, but have not yet explored this. Character-level LSTMs (Kim et al., 2015) may provide robustness to morphology and diacritics and may prove useful even in English for spelling errors and out of vocabulary words. We would also like to investigate how work on unsupervised sense induction (Navigli, 2009) could be integrated with our models to detect out of vocabulary senses – particularly a problem in web text where terminology evolves rapidly.

Finally, many applications of WSD systems for nominal resolution require integration with resolution systems for named entities, since surface forms often overlap (Moro et al., 2014; Navigli and Ponzetto, 2012). This will require inventory alignment work and model reformulation, since we currently use no document-level, topical, or knowledge-base coherence features.

8 Conclusion

In this paper, we have presented several WSD algorithms which combine (1) neural network language models trained on a large unlabeled text corpus, with (2) labeled data in the form of example sentences, and, optionally, (3) unlabeled data in the form of additional sentences. Each of our algorithms looks for the most similar example sentences to the sentence in which the word we wish to classify appears. Using an LSTM language model gave better performance than one based on CBOW word embeddings, although both surpass the current state of the art. The best performance was achieved by our semi-supervised WSD algorithm which builds a graph containing labeled example sentences augmented with a large number of unlabeled sentences from the web, and classifies by propagating sense labels through this graph.

References

[Baroni et al.2014] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247.

[Fellbaum1998] Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

[Gabrilovich and Markovitch2007] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Kim et al.2015] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

[Lesk1986] Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA. ACM.

[Levy and Goldberg2014] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

[Mikolov et al.2011] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *arXiv preprint arXiv:1301.3781*.

[Miller et al.1993] George A Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*, pages 303–308. Association for Computational Linguistics.

[Moro and Navigli2015] Andrea Moro and Roberto Navigli. 2015. Semeval-2015 task 13: multilingual all-words sense disambiguation and entity linking. *Proc. of SemEval*, pages 288–297.

[Moro et al.2014] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

[Navigli and Ponzetto2012] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

[Navigli et al.2007] Roberto Navigli, Kenneth C Litkowski, and Orin Hargraves. 2007. Semeval-2007 task 07: Coarse-grained English all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 30–35. Association for Computational Linguistics.

[Navigli et al.2013] Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. Semeval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 2, pages 222–231.

[Navigli2006] Roberto Navigli. 2006. Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 105–112, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Navigli2009] Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

[Ponzetto and Navigli2010] Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1522–1531, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Pradhan et al.2007] Sameer S Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. Semeval-2007 task 17: English lexical sample, SRL and all words. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 87–92. Association for Computational Linguistics.

[Ravi and Diao2016] Sujith Ravi and Qiming Diao. 2016. Large scale distributed semi-supervised learning using streaming approximation. In *AISTATS*.

[Rothe and Schütze2015] Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China, July. Association for Computational Linguistics.

[Socher et al.2013] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

[Stevenson and Lindberg2010] Angus Stevenson and Christine A. Lindberg. 2010. New Oxford American Dictionary.

[Stevenson2010] Angus Stevenson. 2010. Oxford Dictionary of English.

[Sundermeyer et al.2012] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *INTERSPEECH*, pages 194–197.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

[Taghipour and Ng2015] Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323, Denver, Colorado, May–June. Association for Computational Linguistics.

[Talukdar and Crammer2009] Partha Pratim Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 442–457, Berlin, Heidelberg. Springer-Verlag.

[Turian et al.2010] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

[Zhong and Ng2010] Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL*, ACLDemos '10.