Semi-supervised Word Sense Disambiguation with Neural Models

Dayu Yuan **Julian Richardson** **Ryan Doherty** **Colin Evans** **Eric Altendorf**
Google, Mountain View CA, USA
{dayuyuan,jdcr,portalfire,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. Recently, researchers have shown promising results using word vectors extracted from a neural network language model as features in WSD algorithms. However, a simple average or concatenation of word vectors for each word in a text loses the sequential and syntactic information of the text. In this paper, we study WSD with a sequence learning neural net, LSTM, to better capture the sequential and syntactic patterns of the text. To alleviate the lack of training data in all-words WSD, we employ the same LSTM in a semi-supervised label propagation classifier. We demonstrate state-of-the-art results, especially on verbs.

1 Introduction

Word sense disambiguation (WSD) is a long-standing problem in natural language processing (NLP) with broad applications. 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., 2007a; 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 (Zhong and Ng, 2010), it appears lack of sufficient labeled training data for large vocabularies is the central problem.

One way to leverage unlabeled data is to train a neural network language model (NNLM) on the data. Word embeddings extracted from such a NNLM (often Word2Vec (Mikolov et al., 2013)) can be incorporated as features into a WSD algorithm. Iacobacci et al. (2016) show that this can substantially improve WSD performance and indeed that competitive performance can be attained using word embeddings alone.

In this paper, we describe two novel WSD algorithms. The first is based on a Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). Since this model is able to take into account word order when classifying, it performs significantly better than an algorithm based on a continuous bag of words model (Word2vec) (Mikolov et al., 2013; Iacobacci et al., 2016), especially on verbs.

We then present a semi-supervised algorithm which uses 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.

The best performance was achieved by using an LSTM language model with label propagation. Our algorithm achieves state-of-art performance on many SemEval all-words tasks. It also outperforms the most-frequent-sense and Word2Vec baselines by 10% (see Section 5.2 for details).

Organization: We review related work in Section 2. We introduce our supervised WSD algorithm in Section 3, and the semi-supervised WSD algorithm in Section 4. Experimental results are discussed in Section 5. We provide further discussion and future work in Section 6.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 1374–1385, Osaka, Japan, December 11-17 2016.

2 Related Work

The 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-words 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., 2007a; 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 ~ 20 training examples 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.

A number of different ways have been studied for using word embeddings in WSD. There are some common elements:

- Context embeddings. Given a window of text $w_{n-k}, ..., w_n, ..., w_{n+k}$ surrounding a focus word w_n (whose label is either known in the case of example sentences or to be determined in the case of classification), an embedding for the context is computed as a concatenation or weighted sum of the embeddings of the words $w_i, i \neq n$. Context embeddings of various kinds are used in both (Chen et al., 2014) and (Iacobacci et al., 2016).
- Sense embeddings. Embeddings are computed for each word sense in the word sense inventory (e.g. WordNet). In (Rothe and Schütze, 2015), equations are derived relating embeddings for word senses with embeddings for undisambiguated words. The equations are solved to compute the sense embeddings. In (Chen et al., 2014), sense embeddings are computed first as weighted sums of the embeddings of words in the WordNet gloss for each sense. These are used in an initial bootstrapping WSD phase, and then refined by a neural network which is trained on this bootstrap data.
- Embeddings as SVM features. Context embeddings (Iacobacci et al., 2016; Taghipour and Ng, 2015b), or features computed by combining context embeddings with sense embeddings (Rothe and Schütze, 2015), can be used as additional features in a supervised WSD system e.g. the SVM-based *IMS* (Zhong and Ng, 2010). Indeed Iacobacci et al. (2016) found that using embeddings as the only features in *IMS* gave competitive WSD performance.
- Nearest neighbor classifier. Another way to perform classification is to find the word sense whose sense embedding is closest, as measured by cosine similarity, to the embedding of the classification context. This is used, for example, in the bootstrapping phase of (Chen et al., 2014).
- Retraining embeddings. A feedforward neural network can be used to jointly perform WSD and adjust embeddings (Chen et al., 2014; Taghipour and Ng, 2015b).

In our work, we start with a baseline classifier which uses 1000-dimensional embeddings trained on a 100 billion word news corpus using Word2Vec (Mikolov et al., 2013). The vocabulary consists of the most frequent $1,000,000$ words, without lemmatization or case normalization. Sense embeddings are computed by averaging the context embeddings of sentences which have been labeled with that sense. To classify a word in a context, we assign the word sense whose embedding has maximal cosine similarity with the embedding of the context. This classifier has similar performance to the best classifier in (Iacobacci et al., 2016) when SemCor is used as a source of labeled sentences. The Word2Vec embeddings are trained using a bag of words model, i.e. without considering word order in the training context, and word order is also not considered in the classification context. In Section 3 we show that using a more expressive language model which takes account of word order yields significant improvements.

Semi-supervised learning has previously been applied successfully to word sense disambiguation. In (Yarowsky, 1995) bootstrapping was used to learn a high precision WSD classifier. A low recall classifier was learned from a small set of labeled examples, and the labeled set then extended with those sentences from an unlabeled corpus which the classifier could label with high confidence. The classifier was then retrained, and this iterative training process continued to convergence. Additional heuristics helped to maintain the stability of the bootstrapping process. The method was evaluated on a small data set.

In (Niu et al., 2005), a label propagation algorithm was proposed for word sense disambiguation and compared to bootstrapping and a SVM supervised classifier. Label propagation can achieve better performance because it assigns labels to optimize a *global* objective, whereas bootstrapping propagates labels based on *local* similarity of examples.

In Section 4 we describe our use of label propagation to improve on nearest neighbor for classification.

3 Supervised WSD with LSTM

Neural networks with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) make good language models which take into account word order (Sundermeyer et al., 2012). We train a LSTM language model to predict the held-out word in a sentence. As shown in Figure 1, we first replace the held-out word with a special symbol $, and then, after consuming the remaining words in the sentence, project the h dimensional hidden layer to a p dimensional context layer, and finally predict the held out word with softmax. By default, the LSTM model has 2048 hidden units, 512 dimensional context layer and 512 dimensional word embeddings. We also studied other settings, see Section 5.2.2 for details. We train the LSTM on a news corpus of about 100 billion tokens, with a vocabulary of $1,000,000$ words. Words in the vocabulary are neither lemmatized nor case normalized.

Our LSTM model is different from that of Kgebck and Salomonsson (Kågebäck and Salomonsson, 2016). We train a LSTM language model, which predicts a held-out word given the surrounding context, with a large amount of unlabeled text as training data. The huge training dataset allows us to train a high-capacity model (2048 hidden units, 512 dimensional embeddings), which achieves high precision without overfitting. In our experiments, this directional LSTM model was faster and easier to train than a bidirectional LSTM, especially given our huge training dataset. Kgebck and Salomonsson's LSTM directly predicts the word senses and it is trained with a limited number of word sense-labeled examples. Although regularization and dropout are used to avoid overfitting the training data, the bidirectional LSTM is small with only $74 + 74$ neurons and 100 dimensional word embeddings (Kågebäck and Salomonsson, 2016). Because our LSTM is generally applicable to any word, it achieves high performance on *all-words WSD* tasks (see Section 5 for details), which is the focus of this paper. Kgebck and Salomonsson's LSTM is only evaluated on *lexical sample WSD* tasks of SemEval 2 and 3 (Kågebäck and Salomonsson, 2016).

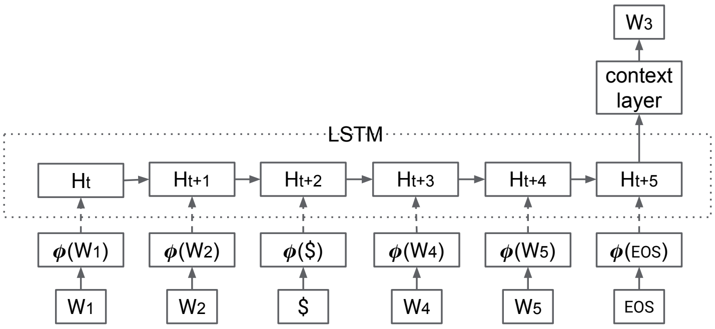


Figure 1: LSTM: Replace the focus word w_3 with a special symbol $ and predict w_3 at the end of the sentence.

The behavior of the LSTM can be intuited by its predictions. Table 1 shows the top 10 words predicted by an LSTM language model for the word 'stock' in sentences containing various senses of 'stock'.

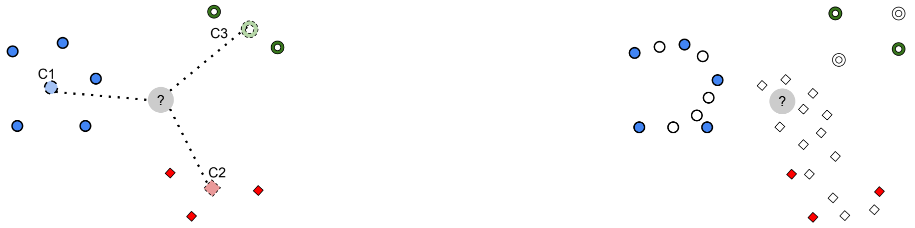
In our initial experiments, we computed similarity between two contexts by the overlap between their bags of predicted words. For example (Table 1) the top predictions for the query overlap most with the LSTM predictions for 'sense#1' —we predict that 'sense#1' is the correct sense. This bag of predictions, 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 context layer from which the bag of predictions was computed as a representation of the context (see Figure 1). Given context vectors extracted from the LSTM, our supervised WSD algorithms classify a word in a context by finding the sense vector which has maximum cosine similarity to the context vector (Figure 2a). We find the sense vectors

id	sentence	top 10 predictions from LSTM	sense
1	Employee compensation is offered in the form of cash and/or *stock*.	cash, stock, equity, shares, loans, bonus, benefits, awards, equivalents, deposits	sense#1
2	The *stock* would be redeemed in five years, subject to terms of the company's debt.	bonds, debt, notes, shares, stock, balance, securities, rest, Notes, debentures	
3	These stores sell excess *stock* or factory overruns .	inventory, goods, parts, sales, inventories, capacity, products, oil, items, fuel	sense#2
4	Our soups are cooked with vegan *stock* and seasonal vegetables.	foods, food, vegetables, meats, recipes, cheese, meat, chicken, pasta, milk	sense#3
query	In addition, they will receive *stock* in the reorganized company, which will be named Ranger Industries Inc.	shares, positions, equity, jobs, awards, representation, stock, investments, roles, funds	?

Table 1: Top predictions of 'stock' in 5 sentences of different word senses

by averaging context vectors of all training sentences of the same sense. We observed in a few cases that the context vector is far from the held-out word's embedding, especially when the input sentence is not informative. For example, the LSTM language model will predict "night" for the input sentence "I fell asleep at [work]." when we hold out "work". Currently, we treat the above cases as outliers. We would like explore alternative solutions, e.g., forcing the model to predict words that are close to one sense vector of the held-out word, in further work. As can be seen in SemEval all-words tasks and Tables 6, this LSTM model has significantly better performance than the Word2Vec models.

4 Semi-supervised WSD



(a) nearest neighbor

(b) semi-supervised WSD with label propagation

Figure 2: WSD classifiers. Filled nodes represent labeled sentences. Unfilled nodes represent unlabeled sentences.

The non-parametric nearest neighbor algorithm described in Section 3 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.

To overcome these drawbacks we present a semi-supervised method which augments the labeled example sentences with a large number of unlabeled sentences from the web. Sense labels are then propagated from the labeled to the unlabeled sentences. Adding a large number of unlabeled sentences allows the decision boundary between different senses to be better approximated.

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 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 with labeled vertices for the labeled example sentences, and unlabeled vertices for sentences containing the lemma, drawn from some additional 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 the 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 2 (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 pairs above the 95 percentile. 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. Our experiments (Table 9) show that this setting achieves good performance on WSD, and the performance is stable when the percentile ranges between 85 to 98. Since it requires running LP for every classification, the algorithm is slow compared to the nearest neighbor algorithm.

5 Experiments

We evaluated the LSTM algorithm with and without label propagation on standard SemEval all-words tasks using WordNet as the inventory. Our proposed algorithms achieve state-of-art performance on many SemEval all-words WSD tasks. In order to assess the effects of training corpus size and language model capacity we also evaluate our algorithms using the New Oxford American Dictionary (NOAD) inventory with SemCor (Miller et al., 1993) or MASC [1].

5.1 SemEval Tasks

In this section, we study the performance of our classifiers on Senseval2 (Edmonds and Cotton, 2001), Senseval3 (Snyder and Palmer, 2004), SemEval-2007 (Pradhan et al., 2007b), SemEval-2013 Task 12 (Navigli et al., 2013) and SemEval-2015 task 13 (Moro and Navigli, 2015) [2]. We focus the study on all-words WSD tasks. For a fair comparison with related works, the classifiers are evaluated on all words (both polysemous and monosemous).

Following related works, we use SemCor or OMSTI (Taghipour and Ng, 2015a) for training. In our LP classifiers, unlabeled data for each lemma consists either of 1000 sentences which contain the lemma, randomly sampled from the web, or all OMSTI sentences (without labels) which contain the lemma.

	Senseval2		Senseval3		SemEval7		SemEval7-Coarse		SemEval13
model	**all**	**n.**	**all**	**n.**	**all**	**n.**	**all**	**n.**	**n.**
IMS + Word2Vec (T:SemCor)	0.634	0.742	0.653	0.701	0.578	0.686			
IMS + Word2Vec (T:OMSTI)	0.683	0.777	0.682	0.741	0.591	0.715			
Taghipour and Ng (2015b)			0.682						
Chen et al. (2014)							0.826	**0.853**	
Weissenborn et al. (2015)				0.688		0.660		**0.855**	**0.728**
Word2Vec (T:SemCor)	0.678	0.737	0.621	0.714	0.585	0.673	0.795	0.814	0.661
LSTM (T:SemCor)	0.736	0.786	0.692	0.723	0.642	**0.723**	0.828	0.834	0.670
LSTM (T:OMSTI)	0.724	0.777	0.643	0.680	0.607	0.673	0.811	0.820	0.673
LSTMLP (T:SemCor, U:OMSTI)	0.739	0.797	0.711	0.748	**0.637**	0.704	**0.843**	0.834	0.679
LSTMLP (T:SemCor, U:1K)	0.738	0.796	**0.718**	**0.763**	0.635	0.717	0.836	0.831	0.695
LSTMLP (T:OMSTI, U:1K)	**0.744**	**0.799**	0.710	0.753	0.633	0.717	0.833	0.825	0.681

Table 2: F1 scores on SemEval all-words tasks. T:SemCor stands for models trained with SemCor. U:OMSTI stands for using OMSTI as unlabeled sentences in semi-supervised WSD. IMS + Word2Vec scores are from (Iacobacci et al., 2016)

Table 2 shows the Sem-Eval results. Our proposed algorithms achieve the highest all-words F1 scores except for Sem-Eval 2013. Weissenborn et al.(2015) only disambiguates nouns, and it outperforms our algorithms on Sem-Eval 2013 by 4%, but is ranked behind our algorithms on Senseval-3 and SemEval-7

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

[2] We mapped all senses to WordNet3.0 by using maps in https://wordnet.princeton.edu/wordnet/download/current-version/ and http://web.eecs.umich.edu/ mihalcea/downloads.html

tasks with an F1 score more than 6% lower than our algorithms. Unified WSD (Chen et al., 2014) has the highest F1 score on Nouns (Sem-Eval-7 Coarse), but our algorithms outperform Unified WSD on other part-of-speech tags.

Settings For a fair comparison of Word2Vec and LSTM, we do not use pre-trained word-embeddings as in (Iacobacci et al., 2016), but instead train the Word2Vec and LSTM models on a 100 billion word news corpus [3] with a vocabulary of the most frequent 1,000,000 words. Our self-trained word embeddings have similar performance to the pre-trained embeddings, as shown in Table 2. The Word2Vec word vectors are of dimension 1024. The LSTM model has 2048 hidden units, and inputs are 512-dimensional word vectors. We train the LSTM model by minimizing sampled softmax loss (Jean et al., 2014) with Adagrad (Duchi et al., 2011). The learning rate is 0.1. We experimented with other learning rates, and observed no significant performance difference after the training converges. We also downsample frequent terms in the same way as (Mikolov et al., 2013).

Word2Vec vectors Vs. LSTM To better compare LSTM with word vectors we also build a nearest neighbor classifier using Word2Vec word embeddings and SemCor example sentences, Word2Vec (T:SemCor). It performs similar to IMS + Word2Vec (T:SemCor), a SVM-based classifier studied in (Iacobacci et al., 2016). Table 2 shows that the LSTM classifier outperforms the Word2Vec classifier across the board.

SemCor Vs. OMSTI Contrary to the results observed in (Iacobacci et al., 2016), the LSTM classifier trained with OMSTI performs worse than that trained with SemCor. It seems that the larger size of the OMSTI training data set is more than offset by noise present in its automatically generated labels. While the SVM classifier studied in (Iacobacci et al., 2016) may be able to learn a model which copes with this noise, our naive nearest neighbor classifiers do not have a learned model and deal less well with noisy labels.

Label propagation We use the implementation of DIST_EXPANDER (Ravi and Diao, 2016). We test the label propagation algorithm with SemCor or OMSTI as labeled data sets and OMSTI or 1000 random sentences from the web per lemma as unlabeled data. The algorithm performs similarly on the different data sets.

Table 3 shows the results of Sem-Eval 2015. The LSTM LP classifier with an LSTM language model achieves the highest scores on nouns and adverbs as well as overall F1. The LSTM classifier has the highest F1 on verbs.

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
Word2Vec (T:SemCor)	0.667	0.661	0.555	0.789	0.810
LSTM (T:SemCor)	0.721	0.713	**0.642**	0.813	0.845
LSTMLP (T:SemCor, U:1K)	**0.726**	**0.728**	0.622	0.813	**0.857**

Table 3: F1 Scores of SemEval-2015 English Dataset. The BFS baseline uses BabelNet first sense.

5.2 NOAD Eval

Many dictionary lemmas and senses have no examples in SemCor or OSTMI, giving rise to losses in all-words WSD when these corpora are used as training data. The above SemEval scores do not distinguish errors caused by missing training data for certain labels or inaccurate classifier. To better study the proposed algorithms, we train the classifiers with the New Oxford American Dictionary (NOAD) (Stevenson and Lindberg, 2010), in which there are example sentences for each word sense.

[3] The training corpus could not be released, but we have plans to open source the well-trained models

5.2.1 Word Sense Inventory

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)[4], provided by Oxford University Press. We manually annotated all words of the English language SemCor corpus and MASC corpora with NOAD word senses in order to evaluate performance [5]. Table 4 shows the total number of polysemes (more than one core sense) and average number of senses per polyseme in NOAD/SELD (hereafter, NOAD), SemCor and MASC. Both SemCor and MASC individually cover around 45% of NOAD polysemes and 62% of senses of those polysemes.

		noun	verb	adj.	adv.
Number of polysemous lemmas in dictionary / corpus	NOAD	8097	2124	2126	266
	SemCor	2833	1362	911	193
	MASC	2738	1250	829	181
Sense count per polyseme	NOAD	2.46	2.58	2.30	2.47
	SemCor	1.44	1.69	1.49	1.84
	MASC	1.48	1.66	1.48	2.01

Table 4: NOAD polysemous lemma in NOAD, SemCor and MASC

Table 5 gives the number of labeled sentences of these datasets. Note that although NOAD has more labeled sentences than SemCor or MASC, the average numbers of sentences per sense of these datasets are similar. This is because NOAD has labeled sentences for each word sense, whereas SemCor (MASC) only covers a subset of lemmas and senses (Table 4). The last column of Table 5 shows that each annotated word in SemCor and MASC has an average of more than 4 NOAD corse senses. Hence, a random guess will have a precision around $1/4$.

	example count (in 1000's)					example count	number of candidate
dataset	all	n.	v.	adj.	adv	per sense	senses per example
NOAD	580	312	150	95	13	18.43	3.1
SemCor	115	38	57	11.6	8.6	14.27	4.1
MASC	133	50	57	12.7	13.6	17.38	4.2

Table 5: Number of examples in each dataset and the average sense count per example.

In the default setting, we use NOAD example sentences as labeled training data and evaluate on SemCor and MASC. We evaluate all polysemous words in the evaluation corpus.

5.2.2 LSTM classifier

We compare our algorithms with two baseline algorithms:

- Most frequent sense: Compute the sense frequency (from a labeled corpus) and label word w with w's most frequent sense.
- Word2Vec: a nearest-neighbor classifier with Word2Vec word embedding, which has similar performance to cutting-edge algorithms studied in (Iacobacci et al., 2016) on SemEval tasks.

Table 6 compares the F1 scores of the LSTM and baseline algorithms. LSTM outperforms Word2Vec 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 Word2Vec, is important to distinguishing word senses of verbs and adverbs.

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

[5] https://research.google.com/research-outreach.html#/research-outreach/research-datasets

eval data		SemCor					MASC				
model	**train data**	**all**	**n.**	**v.**	**adj.**	**adv.**	**all**	**n.**	**v.**	**adj.**	**adv.**
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					
Word2Vec	NOAD	0.709	0.783	0.657	0.736	0.693	0.671	0.790	0.562	0.724	0.638
Word2Vec	SemCor						0.692	0.806	0.592	0.754	0.635
Word2Vec	NOAD,SemCor						0.678	0.808	0.565	0.753	0.604
Word2Vec	MASC	0.698	0.785	0.619	0.766	0.744					
Word2Vec	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					

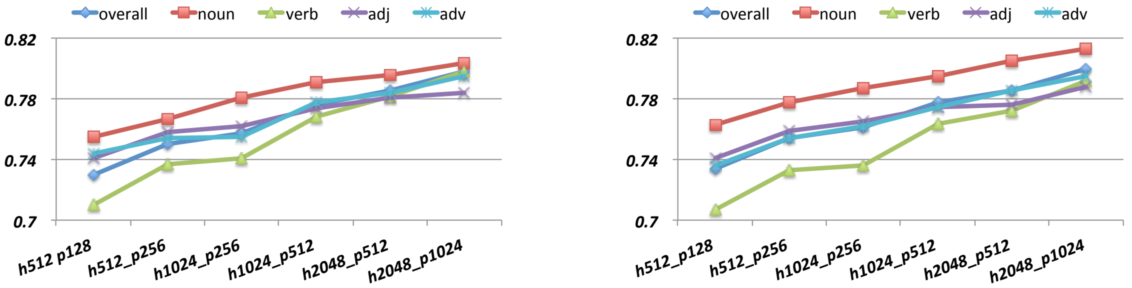
Table 6: F1 scores of LSTM algorithm in comparison with baselines

eval data		SemCor					MASC				
model	**train data**	**all**	**n.**	**v.**	**adj.**	**adv.**	**all**	**n.**	**v.**	**adj.**	**adv.**
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**					

Table 7: Macro F1 scores of LSTM classifier

Change of 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 6. Across all part of speech tags and datasets, 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, as shown in Table 7, because of the limited coverage of rare senses and words in SemCor and MASC.

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 3 shows strong positive 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.



(a) eval on SemCor

(b) eval on MASC

Figure 3: F1 scores of LSTM models with different capacity: h is the number of hidden units; p is the embedding dimension.

5.2.3 Semi-supervised WSD

We evaluate our semi-supervised WSD classifier in this subsection. We construct the graph as described in Section 4 and run LP to propagate sense labels from the seed vertices to the unlabeled vertices. We evaluate the performance of the algorithm by comparing the predicted labels and the gold labels on eval nodes. As can be observed from Table 8, LP did not yield clear benefits when using the Word2Vec language model. We did see significant improvements, 6.3% increase on SemCor and 7.3% increase on MASC, using LP with the LSTM language model. We hypothesize that this is because LP is sensitive to the quality of the graph distance metric.

eval data		SemCor					MASC				
model	**train data**	**all**	**n.**	**v.**	**adj.**	**adv.**	**all**	**n.**	**v.**	**adj.**	**adv.**
Word2Vec LP	NOAD	0.642	0.733	0.554	0.705	0.725	0.643	0.752	0.524	0.726	0.664
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 8: F1 scores of label propagation

Change of seed data: As can be seen in Table 8, LP substantially improves classifier F1 when the training datasets are SemCor+NOAD or MASC+NOAD. As discussed in Section 4, the improvement may come from explicitly modeling the sense prior. We did not see much performance lift by increasing the number of unlabeled sentences per lemma.

Change of graph density: By default, we construct the LP graph by connecting two nodes if their affinity is above 95% percentile. We also force each node to connect to at least 10 neighbors to prevent isolated nodes. Table 9 shows the performance of the LP algorithm by changing the percentile threshold. The F1 scores are relatively stable when the percentile ranges between 85 to 98, but decrease when the percentile drops to 80. Also, it takes longer to run the LP algorithm on a denser graph. We pick the 95 percentile in our default setting to achieve both high F1 scores and short running time.

	SemCor						MASC					
pos-tag	**98**	**95**	**90**	**85**	**80**	**70**	**98**	**95**	**90**	**85**	**80**	**70**
overall	**0.823**	0.822	**0.823**	0.818	0.813	0.800	0.827	0.831	**0.835**	0.830	0.824	0.806
n.	0.848	**0.859**	0.852	0.846	0.840	0.828	0.863	0.865	**0.868**	0.865	0.861	0.847
v.	**0.810**	0.800	0.803	0.797	0.792	0.778	0.800	**0.806**	**0.806**	0.799	0.794	0.769

Table 9: F1 scores of the LSTM LP trained on NOAD with varying graph density.

6 Conclusions and Future Work

In this paper, we have presented two WSD algorithms which combine (1) LSTM 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. Using an LSTM language model gave better performance than one based on Word2Vec embeddings. 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.

Several unanswered questions suggest lines of future work. 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 like to see whether our results can be improved by incorporating global (document) context and multiple embeddings for polysemous words (Huang et al., 2012).

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.

We thank our colleagues and the anonymous reviewers for their insightful comments on this paper.

References

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*, pages 1025–1035. Citeseer.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France, July. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

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

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *ACL*. Citeseer.

S. Jean, K. Cho, R. Memisevic, and Y. Bengio. 2014. On Using Very Large Target Vocabulary for Neural Machine Translation. *ArXiv e-prints*, December.

M. Kågebäck and H. Salomonsson. 2016. Word Sense Disambiguation using a Bidirectional LSTM. *ArXiv e-prints*.

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

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

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.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

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.

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

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.

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.

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.

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.

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.

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

Zheng-Yu Niu, Dong-Hong Ji, and Chew Lim Tan. 2005. Word sense disambiguation using label propagation based semi-supervised learning. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 395–402. Association for Computational Linguistics.

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.

Sameer S Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007a. 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.

Sameer S Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007b. 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.

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

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.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43.

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.

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

Angus Stevenson. 2010. Oxford Dictionary of English.

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

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.

Kaveh Taghipour and Hwee Tou Ng. 2015a. One million sense-tagged instances for word sense disambiguation and induction. *CoNLL 2015*, page 338.

Kaveh Taghipour and Hwee Tou Ng. 2015b. 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.

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.

Dirk Weissenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. 2015. Multi-objective optimization for the joint disambiguation of nouns and named entities. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 596–605.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics.

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