A Deep Dive into Word Sense Disambiguation with LSTM

Minh Le[†], **Marten Postma**[†], **Jacopo Urbani**[‡] and **Piek Vossen**[†]
[†] Department of Language, Literature and Communication, Vrije Universiteit Amsterdam
[‡] Department of Computer Science, Vrije Universiteit Amsterdam
`{m.n.le,m.c.postma,piek.vossen}@vu.nl`, `jacopo@cs.vu.nl`

Abstract

LSTM-based language models have been shown effective in Word Sense Disambiguation (WSD). In particular, the technique proposed by Yuan et al. (2016) returned state-of-the-art performance in several benchmarks, but neither the training data nor the source code was released. This paper presents the results of a reproduction study and analysis of this technique using only openly available datasets (GigaWord, SemCor, OMSTI) and software (TensorFlow). Our study showed that similar results can be obtained with much less data than hinted at by Yuan et al. (2016). Detailed analyses shed light on the strengths and weaknesses of this method. First, adding more unannotated training data is useful, but is subject to diminishing returns. Second, the model can correctly identify both popular and unpopular meanings. Finally, the limited sense coverage in the annotated datasets is a major limitation. All code and trained models are made freely available.

1 Introduction

Word Sense Disambiguation (WSD) is a long-established task in the NLP community (see Navigli (2009) for a survey) which goal is to annotate lemmas in text with the most appropriate meaning from a lexical database like WordNet (Fellbaum, 1998). Many approaches have been proposed – the more popular ones include the usage of Support Vector Machine (SVM) (Zhong and Ng, 2010), SVM combined with unsupervised trained embeddings (Iacobacci et al., 2016; Rothe and Schütze, 2017), and graph-based approaches (Agirre et al., 2014; Weissenborn et al., 2015).

In recent years, there has been a surge in interest in using Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) to perform WSD (Raganato et al., 2017b; Melamud et al., 2016). These approaches are characterized by their high performance, simplicity and their ability to extract a lot of information from raw text. Among the best-performing ones is the approach by Yuan et al. (2016), in which an LSTM language model trained on a corpus with 100 billion tokens was coupled with small sense-annotated datasets to achieve state-of-the-art performance in all-words WSD.

Even though the results obtained by Yuan et al. (2016) outperform the previous state-of-the-art, neither the used datasets nor the constructed models are available to the community. This is unfortunate because this makes the re-application of this technique a non-trivial process, and it hinders further studies for understanding which limitations prevent even higher accuracies. These could be, for instance, of algorithmic nature or relate to the input (either size or quality), and a deeper understanding is crucial for enabling further improvements. In addition, some details are not reported, and this could prevent other attempts from replicating the results.

To address these issues, we reimplemented Yuan et al. (2016)'s method with the goal of: 1) reproducing and making available the code, trained models, and results and 2) understanding which are the main factors that constitute the strengths and weaknesses of this method. While a full replication is not possible due to the unavailability of the original data, we nevertheless managed to reproduce their approach with other public text corpora, and this allowed us to perform a deeper investigation on the performance of this technique. This investigation aimed at understanding how sensitive the WSD approach is w.r.t. the amount of unannotated data (i.e., raw text) used for training, model complexity, how biased the method is towards the choice of the most frequent senses (MFS), and identifying limitations that cannot be overcome with bigger unannotated datasets.

The contribution of this paper is thus two-fold: On the one hand, we present a reproduction study whose results are publicly available and hence can be freely used by the community. Notice that the lack of available models has been explicitly mentioned, in a recent work, as the cause for the missing comparison of this technique with other competitors (Raganato et al., 2017b, footnote 10). On the other hand, we present other experiments to shed more light on the value of this and similar methods.

We anticipate some conclusions. First, a positive result is that we were able to reproduce the method from Yuan et al. (2016) and obtain similar results to the ones originally published. However, to our surprise, these results were obtained using a much smaller corpus of 1.8 billion tokens (Gigaword), which is less than 2% of the data used in the original study. In addition, we observe that the amount of unannotated data is important, but that the relationship between its size and the improvement is not linear, meaning that exponentially more unannotated data is needed in order to improve the performance. Moreover, we show that the percentage of correct sense assignments is more balanced w.r.t sense popularity, meaning that the system has a less-strong bias towards the most-frequent sense (MFS) and is better at recognizing both popular and unpopular meanings. Finally, we show that the limited sense coverage in the annotated datasets is a major limitation, as shown by the fact that resulting model does not have a representation for more than 30% of the meanings which should have been considered for disambiguating the test sets.

2 Background

Current WSD systems can be categorized according to two dimensions: whether they use raw text without any preassigned meaning (unannotated data henceforth), and whether they exploit the relations between synsets in WordNet (synset relations henceforth). One prominent state-of-the-art system that does not rely on unannotated data nor exploits synset relations is It Makes Sense (IMS) (Zhong and Ng, 2010; Taghipour and Ng, 2015). This system uses an SVM to train classifiers for each lemma using only annotated data as training evidence.

In contrast, graph-based WSD systems do not use (un)annotated data but rely on the synset relations. The system UKB (Agirre et al., 2014) represents WordNet as a graph where the synsets are the nodes and the relations are the edges. After the node weights have been initialized using the Personalized Page Rank algorithm, they are updated depending on context information. Then, the synset with the highest weight is chosen. Babelfy (Moro et al., 2014) and the system by Weissenborn et al. (2015) both represent the whole input document as a graph with synset relations as edges and jointly disambiguate nouns and verbs. In the case of Babelfy, a densest-subgraph heuristic is used to compute the high-coherence semantic interpretations of the text. Instead, in Weissenborn et al. (2015) a set of complementary objectives, which include sense probabilities and type classification, are combined together to perform WSD.

A number of systems make use of both unannotated data and synset relations. Both Tripodi and Pelillo (2017) and Camacho-Collados et al. (2016) make use of statistical information from unannotated data to weigh the relevance of nodes in a graph, which is then used to perform WSD. Rothe and Schütze (2017) use word embeddings as a starting point and then rely on the formal constraints in a lexical resource to create synset embeddings.

Recently, there has been a surge in WSD approaches that use unannotated data but do not consider synset relations. One example is provided by Iacobacci et al. (2016), who investigated the role of word embeddings as features in a WSD system. Four methods (concatenation, average, fractional decay, and exponential decay) are used to extract features from the sentential context using word embeddings. The features are then added to the default feature set of IMS (Zhong and Ng, 2010). Moreover, Raganato et al. (2017b) present a number of end-to-end neural WSD architectures. The best performing one is based on a bidirectional Long Short-Term Memory (BLSTM) with attention and two auxiliary loss functions (part-of-speech and the WordNet coarse-grained semantic labels). Melamud et al. (2016) also make use of unannotated data to train a BLSTM. The work by Yuan et al. (2016), which we consider in this paper,

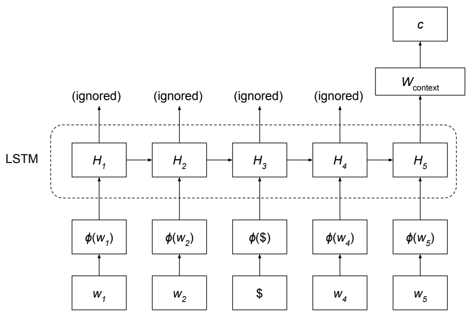


Figure 1: The LSTM model used to perform language modeling and compute context embeddings. At training time, a softmax layer is added, allowing it to predict the omitted word; at test time, the context embeddings are used for WSD in a nearest-neighbor or label-propagation procedure.

belongs to this last category. Different from Melamud et al. (2016), it uses significantly more unannotated data, the model contains more hidden units (2048 vs. 600), and the sense assignment is more elaborated. We describe this approach in more detail in the following section.

3 WSD with Language Models

The method proposed by Yuan et al. (2016) performs WSD by annotating each lemma in a text with one WordNet synset that is associated with its meaning. Broadly speaking, the disambiguation is done by: 1) constructing a language model from a large unannotated dataset; 2) extracting sense embeddings from this model using a much smaller annotated dataset; 3) relying on the sense embeddings to make predictions on the lemmas in unseen sentences. Each operation is described below.

Constructing Language Models. Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a celebrated recurrent neural network architecture that has proven to be effective in many natural language processing tasks (Sutskever et al., 2014; Dyer et al., 2015; He et al., 2017, among others). Different from previous architectures, LSTM is equipped with trainable gates that control the flow of information, allowing the neural networks to learn both short- and long-range dependencies.

In Yuan et al. (2016), the first operation consists of constructing an LSTM language model to capture the meaning of words in context. They use an LSTM network with a single hidden layer of h nodes. Given a sentence $s = (w_1, w_2, \dots, w_n)$, they replace word $w_k \, (1 \leq k \leq n)$ by a special token $\$$. The model takes this new sentence as input and produces a context vector $\mathbf{c}$ of dimensionality p (see Figure 1).[1]

Each word w in the vocabulary $\mathcal{V}$ is associated with an embedding $\phi_\text{o}(w)$ of the same dimensionality. The model is trained to predict the omitted word, minimizing the softmax loss over a big collection $\mathcal{D}$ of sentences.

$$\ell = -\sum_{s \in \mathcal{D}} \sum_{k=1}^{|s|} \log \frac{\exp(\mathbf{c} \cdot \phi_\text{o}(w_k))}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{c} \cdot \phi_\text{o}(w'))}$$

After the model is trained, we can use it to extract *context* embeddings, i.e., latent numerical representations of the sentence surrounding a given word.

Calculating Sense Embeddings. The model produced by the LSTM network is meant to capture the "meaning" of words in the context they are mentioned. In order to perform the sense disambiguation, we need to extract from it a suitable representation for word senses. To this purpose, the method relies on another corpus where each word is annotated with the corresponding sense.

The main intuition is that words used with the same sense are mentioned in contexts which are very similar to each other as well. This suggests a simple way to calculate sense embeddings. First, the LSTM model is invoked to compute the context vector for each occurrence of one sense in the annotated dataset. Once all context vectors are computed, the sense embedding is defined as the average of all vectors. Let

[1]As usual, vectors are indicated with boldface to distinguish them to scalar and other symbols.

us assume, for instance, that the sense $horse_n^2$ (that is, the second sense of horse as a noun) appears in the two sentences:

(1) The move of the $horse_n^2$ to the corner forced the checkmate.

(2) Karjakin makes up for his lost bishop a few moves later, trading rooks and winning black's $horse_n^2$.

In this case, the method will replace the sense by \$ in the sentences and feed them to the trained LSTM model to calculate two context vectors $\mathbf{c_1}$ and $\mathbf{c_2}$. The sense embedding $\mathbf{s_{horse_n^2}}$ is then computed as:

$$\mathbf{s_{horse_n^2}} = \frac{\mathbf{c_1} + \mathbf{c_2}}{2}$$

This procedure is computed for every sense that appears in the annotated corpus.

Averaging technique to predict senses. After all sense embeddings are computed, the method is ready to disambiguate target words. This procedure proceeds as follows:

1. Given an input sentence and a target word, it replaces the occurrence of the target word by \$ and uses the LSTM model to predict a context vector $\mathbf{c_t}$.
2. The lemma of the target word is used to retrieve from WordNet the candidate synsets $s_1, \ldots, s_n$ where n is the number of synsets. Then, the procedure looks up the corresponding sense embeddings $\mathbf{s_1}, \ldots, \mathbf{s_n}$ computed in the previous step.
3. The procedure invokes a subroutine to choose one of the n senses for the context vector $\mathbf{c_t}$. It selects the sense whose vector is closest to $\mathbf{c_t}$ using cosine as the similarity function.

Label Propagation. Yuan et al. (2016) argue that the averaging procedure is suboptimal because of two reasons. First, the distribution of occurrences of senses is unknown whereas averaging is only suitable for spherical clusters. Second, averaging reduces the representation of occurrences of each sense to a single vector and therefore ignores sense prior. For this reason, they propose to use label propagation for inference as an alternative to averaging. Label propagation (Zhu and Ghahramani, 2002) is a classic semi-supervised algorithm that has been employed in WSD (Niu et al., 2005) and other NLP tasks (Chen et al., 2006; Zhou, 2011). The procedure involves predicting senses for not only the target cases but also for unannotated words queried from a corpus. It represents both the target cases and unannotated words as points in a vector space and iteratively propagates classification labels from the target classes to the words. In this way, it can be used to construct non-spherical clusters and to give more influence to frequent senses.

Overall algorithm. The overall disambiguation procedure that we implemented proceeds as follows:

1. *Monosemous:* First, the WSD algorithm checks whether the target lemma is monosemous (i.e., there is only one synset). In this case, the disambiguation is trivial.
2. *Label propagation:* If the label propagation is enabled, then it checks whether the target lemma occurs at least once in the annotated dataset and at least once in the auxiliary unannotated dataset. In this case, the procedure applies the *label propagation* technique for selecting the candidate synset.
3. *Averaging:* If the previous strategies are not applicable and there is at least one occurrence of the target lemma in the annotated dataset, then we apply the *averaging* technique for selecting the candidate synset.
4. *MFS fallback:* If the target lemma does not appear in the annotated dataset, then the system picks the most-frequent synset.[2]

[2]Notice that Yuan et al. (2016) did not report if they use an MFS-fallback strategy or simply returned no answer.

4 Reproduction Study: Methodology

Before we report the results of our experiments, we describe the datasets used and give some details regarding our implementation.

Training data. The 100-billion-token corpus used in the original publication is not publicly available. Therefore, for the training of the LSTM models, we used the English Gigaword Fifth Edition (Linguistic Data Consortium (LDC) catalog number LDC2011T07). The corpus consists of 1.8 billion tokens in 4.1 million documents, originated from four major news agencies. We leave the study of bigger corpora for future work.

For the training of the sense embeddings, we use the same two corpora used by Yuan et al. (2016):

1. *SemCor* (Miller et al., 1993) is a corpus containing approximately 240,000 sense annotated words. The tagged documents originate from the Brown corpus (Francis and Kucera, 1979) and cover various genres.
2. *OMSTI* (Taghipour and Ng, 2015) contains one million sense annotations automatically tagged by exploiting the English-Chinese part of the parallel MultiUN corpus (Eisele and Chen, 2010). A list of English translations were manually created for each WordNet sense. If the Chinese translation of an English word matches one of the manually curated translations for a WordNet sense, that sense is selected.

Implementation. We used the BeautifulSoup HTML parser to extract plain text from the Gigaword corpus. Then, we used the English models[3] of Spacy 1.8.2 for sentence boundary detection and tokenization. The LSTM model is implemented using TensorFlow 1.2.1 (Abadi et al., 2015). We chose TensorFlow because of its industrial-grade quality and because it can train large-scale models.

The main computational bottleneck of the entire process is the training of the LSTM model. Although we do not use a 100-billion-token corpus, training the model on Gigaword can already take years if not optimized properly. To reduce training time, we assumed that all (padded) sentences in the batch have the same length. This optimization increases the speed by 17% as measured on a smaller model ($h = 100, p = 10$). Second, following Yuan et al., we use the sampled softmax loss function (Jean et al., 2015). Third, we grouped sentences of similar length together while varying the number of sentences in a batch to fully utilize GPU RAM. Together, these heuristics increased training speed by 42 times.

Although Yuan et al. proposed to use a distributed implementation of label propagation (Ravi and Diao, 2015), we found that scikit-learn (Pedregosa et al., 2011) was fast enough for our experiments. For hyperparameter tuning, we use the annotations in OMSTI (which are not used at test time). After measuring the performance of some variations of label propagation (scikit-learn implementation: *LabelPropagation* or *LabelSpreading*; similarity measure: *inner product* or radial basis function with different values of γ), we found that the combination of *LabelSpreading* and *inner product* similarity leads to the best result which is also better than averaging on the development set.

Evaluation framework. For evaluating the WSD predictions, we selected two test sets: one from the Senseval2 (Palmer et al., 2001) competition, which tests the disambiguation of nouns, verbs, adjectives and adverbs, and one from the 2013 edition (Navigli et al., 2013), which focuses only on nouns.

The test set from Senseval-2 is the English All-Words Task; *senseval2* henceforth. This dataset contains 2,282 annotations from three articles from the Wall Street Journal. Most of the annotations are nominal, but the competition also contains annotations for verbs, adjectives, and adverbs. In this test set, 66.8% of all target words are annotated with the most-frequent sense (MFS) of the lemma. This means that the simple strategy of always selecting the MFS would score 66.8% F_1 on this dataset.

The test set from SemEval-2013 is the one taken from task 12: Multilingual Word Sense Disambiguation; *semeval2013* henceforth. This task consists of two disambiguation tasks: Entity Linking and Word Sense Disambiguation for English, German, French, Italian, and Spanish. This test set contains 13 articles from previous editions of the workshop on Statistical Machine Translation.[4] The articles contain

[3] `en_core_web_md-1.2.1`

[4] `http://www.statmt.org`

Model/Method	Datasets	Scorer	senseval2 F_1	semeval2013 F_1
Our LSTM	T: SemCor	framework	**0.720**	0.647
Our LSTM	T: OMSTI	framework	0.675	0.651
Our LSTM	T: SemCor+OMSTI	framework	0.699	**0.656**
Our LSTMLP	T: SemCor, U:OMSTI	framework	0.714	0.642
Yuan et al. (2016) LSTM	T: SemCor	mapping to WN3.0	0.736	0.670
Yuan et al. (2016) LSTM	T: OMSTI	mapping to WN3.0	0.724	0.673
Yuan et al. (2016) LSTMLP	T: SemCor, U: OMSTI	mapping to WN3.0	**0.739**	**0.679**
Raganato et al. (2017b)	T: SemCor	framework	0.720	0.669
Iacobacci et al. (2016) IMS+emb	T: SemCor	framework	0.710	0.673
Iacobacci et al. (2016) IMS+emb	T: SemCor+OMSTI	framework	0.708	0.663
Iacobacci et al. (2016) IMS-s+emb	T: SemCor	framework	0.722	0.659
Iacobacci et al. (2016) IMS-s+emb	T: SemCor+OMSTI	framework	**0.733**	0.667
Melamud et al. (2016)	T: SemCor	framework	0.718	0.656
Melamud et al. (2016)	T: SemCor+OMSTI	framework	0.723	0.672
Agirre et al. (2014) UKB-g*	P: SemCor	framework	0.688	0.688
Weissenborn et al. (2015)	P: SemCor	competition	-	**0.728**

Table 1: Performance of our implementation compared to already published results. We report the model/method used to perform WSD, the used annotated dataset and scorer, and F_1 for each test set. In the naming of our models, *LSTM* indicates that the *averaging* technique was used for the sense assignment, while *LSTMLP* refers to the results obtained using *label propagation* (see Section 3). The datasets following *T:* indicate the annotated corpus used to represent the senses while *U:OMSTI* stands for using OMSTI as unlabeled sentences in case label propagation is used. *P: SemCor* indicates that sense distributions from SemCor are used in the system architecture. Three scorers are used: *"framework"* refers to the WSD evaluation framework from Raganato et al. (2017a); *"mapping to WN3.0"* refers to the evaluation used by Yuan et al. (2016) while *"competition"* refers to the scorer provided by the competition itself (e.g., semeval2013).

1,644 test instances in total, which are all nouns. The application of the MFS baseline on this dataset yields an F_1 score of 63.0%.

5 Results

In this section, we report our reproduction of the results of Yuan et al. (2016) and additional experiments to gain a deeper insight into the strengths and weaknesses of the approach. These experiments focus on the performance on the most- and less-frequent senses, coverage of the annotated dataset and the consequent impact on the overall predictions, the granularity of the sense representation, and the impact of the unannotated data and model complexity on the accuracy of WSD.

Reproduction results. We trained the LSTM model with the best reported settings in Yuan et al. (2016) (hidden layer size $h = 2048$, embedding dimensionality $p = 512$) using a machine equipped with an Intel Xeon E5-2650, 256GB of RAM, 8TB of disk space, and two nVIDIA GeForce GTX 1080 Ti GPUs. During our training, one epoch took about one day to finish with TensorFlow fully utilizing one GPU. The whole training process took four months. We tested the performance of the downstream WSD task three times during the training and observed that the best performance is obtained at the 65th epoch, despite a later model producing a lower negative log-likelihood. Thus, we used the model produced at the 65th epoch for our experiments below.

Table 1 presents the results using the test sets *senseval2* and *semeval2013*, respectively. The top part of the table presents our reproduction results, the middle part reports the results from Yuan et al. (2016), while the bottom part reports a representative sample of the other state-of-the-art approaches.

It should be noted that with the test set *semeval2013*, all scorers use WordNet 3.0, therefore the performance of the various methods can be directly compared. However, not all answers in *senseval2* can be mapped to WN3.0 and we do not know how Yuan et al. (2016) handled these cases. In the WSD evaluation framework (Moro et al., 2014) that we selected for evaluation, these cases were either re-annotated or removed. Thus, our F_1 on *senseval2* cannot be directly compared with the F_1 in the original paper.

From a first glance at Table 1, we observe that if we use SemCor to train the synset embeddings, then our results come close to the state-of-the-art on *senseval2* (0.720 vs. 0.733). On *semeval2013*, we achieve results comparable to other embeddings-based approaches (Raganato et al., 2017b; Iacobacci et

Model	senseval2 F_1	senseval2 R_mfs (n=1524)	senseval2 R_lfs (n=758)	semeval2013 F_1	semeval2013 R_mfs (n=1035)	semeval2013 R_lfs (n=609)
Our LSTM (T: SemCor)	**0.72**	0.88	**0.41**	0.65	0.84	0.33
Our LSTM (T: OMSTI)	0.67	0.87	0.27	0.65	**0.86**	0.29
Our LSTM (T: SemCor+OMSTI)	0.70	0.85	0.40	**0.66**	0.82	**0.38**
Our LSTMLP (T: SemCor+OMSTI)	0.71	**0.91**	0.32	0.64	0.85	0.29

Table 2: Performance of our implementation with respect to MFS and LFS recall. *R_mfs* and *R_lfs* are the recall on the most-frequent-sense and least-frequent-sense instances respectively. n represents the number of considered instances.

al., 2016; Melamud et al., 2016). However, the gap with the graph-based approach of Weissenborn et al. (2015) is still significant. When we use both SemCor and OMSTI for the annotated data, our results drop 0.02 point for *senseval2*, whereas they increase by almost 0.01 for *semeval2013*. Different from Yuan et al. (2016), we did not observe improvement by using label propagation (comparing *T: SemCor, U: OMSTI* against *T:SemCor* without propagation). However, the performance of the label propagation strategy is still competitive on both test sets.

Most- vs. less-frequent-sense instances. The original paper only analyses the performance on the whole test sets. We extend this analysis by looking at the performance for disambiguating the most frequent-sense (MFS) and less-frequent-sense (LFS) instances. The first type of instances are the ones for which the correct link is the most-frequent sense, whereas the second subset consists of the remaining ones. This analysis is important because it is well-known that the simple strategy of always choosing the MFS is a strong baseline in WSD, thus there is a tendency for WSD systems to overfit towards the MFS (Postma et al., 2016).

Table 2 shows that the method by Yuan et al. (2016) does not overfit towards the MFS to the same extent as other supervised systems since the recall on LFS instances is still quite high 0.41 (a lower recall on LFS instances than on MFS ones is expected due to the reduced training data for them).

On semeval13, the recall on LFS is already relatively high using only SemCor (0.33), and reaches 0.38 when using both SemCor and OMSTI. For comparison, the default system IMS (Zhong and Ng, 2010) trained on SemCor only obtains an *R_lfs* of 0.15 on semeval13 (Postma et al., 2016) and only reaches 0.33 with a large amount of annotated data.

Finally, our implementation of the label propagation does seem to slightly overfit towards the MFS. When we compare the results of the *averaging technique* using SemCor and OMSTI versus when we use *label propagation*, we notice an increase in the MFS recall (from 0.85 to 0.91), whereas the LFS recall drops from 0.40 to 0.32.

Meaning coverage in annotated datasets. The WSD procedure depends on an annotated corpus to compose its sense representations, making missing annotations an insurmountable obstacle. In fact, annotated datasets only contain annotations for a proper subset of the possible candidate synsets listed in WordNet. We analyze this phenomenon using four statistics:

1. *Candidate Coverage:* For each test set, we performed a lookup in WordNet to determine the unique candidate synsets of all target lemmas. We then determined what percentage of these candidate synsets that have *at least one* annotation in the annotated dataset.
2. *Lemma Coverage:* Given a target lemma in a test set, we performed a lookup in WordNet to determine the unique candidate synsets. If *all* candidate synsets of that target lemma have at least one annotation in the annotated dataset, we claim that the lemma is covered. The lemma coverage is then the percentage of all covered target lemmas. A high lemma coverage indicates that annotated dataset covers most of the meanings in the test set.
3. *Gold Coverage:* We calculate the percentage of the *correct* answers in the test set that has at least one annotation in the annotated dataset.

The column "Candidate Coverage" of Table 3 shows that SemCor only contains less than 70% of all candidate synsets for *senseval2* and *semeval2013*, meaning that a model will never have a representation

Datasets	senseval2 Candidate Coverage	senseval2 Lemma Coverage	senseval2 Gold Coverage	semeval2013 Candidate Coverage	semeval2013 Lemma Coverage	semeval2013 Gold Coverage
SemCor	63.51%	29.72%	80.92%	66.64%	28.42%	83.08%
OMSTI	29.98%	5.9%	43.82%	31.22%	5.94%	45.23%
SemCor+OMSTI	**66.26%**	**32.25%**	**81.98%**	**69.89%**	**31.83%**	**84.76%**

Table 3: Statistics about the coverage of annotated datasets in WordNet.

Competition	Model	MFS fallback	Averaging	Label propagation
senseval2	Our LSTM (T: SemCor)	0.64 (n=135)	0.66 (n=1712)	-
	Our LSTM (T: OMSTI)	0.66 (n=775)	0.56 (n=1072)	-
	Our LSTM (T: SemCor+OMSTI)	0.64 (n=135)	0.63 (n=1712)	-
	Our LSTMLP (T:SemCor, U:OMSTI)	0.64 (n=135)	0.71 (n=644)	0.61 (n=1068)
semeval2013	Our LSTM (T: SemCor)	0.39 (n=41)	0.56 (n=1262)	-
	Our LSTM (T: OMSTI)	0.58 (n=427)	0.55 (n=876)	-
	Our LSTM (T: SemCor+OMSTI)	0.40 (n=40)	0.57 (n=1263)	-
	Our LSTMLP (T:SemCor, U:OMSTI)	0.40 (n=40)	0.57 (n=397)	0.55 (n=866)

Table 4: The recall is shown for three WSD strategies and n indicates the number of instances it was actually used. The recall on all monosemous instances was 1.0 for both competitions and is hence not shown in the table. There are 435 monosemous instances in *senseval2* and 341 in *semeval2013*.

for more than 30% of the candidate synsets. Even with the addition of OMSTI, the coverage does not exceed 70%, meaning that we lack evidence for a significant number of potential annotations. Moreover, the column "Lemma Coverage" illustrates that we have evidence for all potential solutions for only 30% of the lemmas in both WSD competitions, meaning that in the large majority of the cases some solutions are never seen. The column "Gold coverage" measures whether the right answers are at least seen in the annotated dataset. The numbers illustrate that 20% of the solutions in the test sets do not have any annotations. With our approach, these answers can only be returned if the lemma is monosemous or by random guess otherwise.

To further investigate these issues, Table 4 reports the recall of the various disambiguation strategies which could be invoked depending on the coverage of the lemma (these can be: *monosemous*, *averaging*, *label propagation*, *MFS* – see the overall procedure reported in Section 3).

We observe that the MFS fallback plays a significant role in obtaining the overall high accuracy since it is invoked many times, especially with OMSTI due to the low coverage of the dataset (in this case it is invoked in 775 cases vs. 1072 of averaging). For example, if we had not applied the MFS fallback strategy for *senseval2* using SemCor as the annotated corpus, our performance would have dropped from 0.72 to 0.66, below the MFS baseline of 0.67 for this task.[5] Label propagation was indeed applied on half of the cases, but leads to lower results. From these results, we learn that the effectiveness of this method strongly depends on the coverage of the annotated datasets: If it is not high, as it is with OMSTI, then the performance of this method reduces to the one of choosing the MFS.

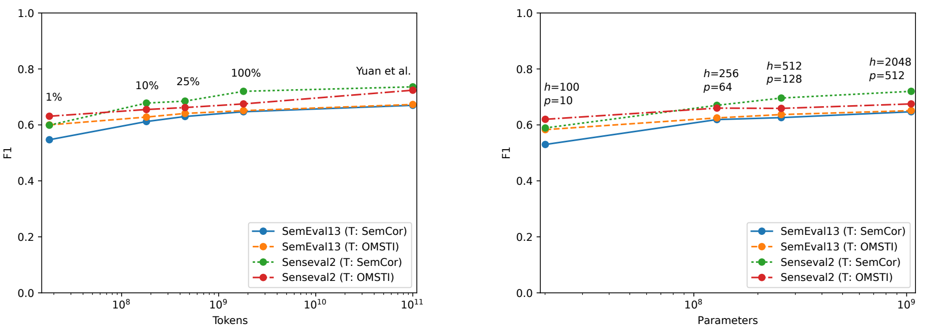
Granularity of sense representation. Rothe and Schütze (2017) provided evidence for the claim that the granularity of the sense representations has an influence on WSD performance. More in particular, their WSD system performed better when trained on sensekeys (called *lexemes* in their paper) than on synsets. Although a sensekey-based disambiguation results in less annotated data per target lemma, the sensekey representation is more precise (since it is a lemma associated with a particular meaning) than at the synset level.

The reimplementation discussed in this paper allows us to answer the question: "How will LSTM models work if we lower the disambiguation level from synset to sensekey?" Table 5 presents the results of this experiment. As we can see from the table, our method also returns better performance on both test sets. This behavior is interesting and one possible explanation is that sensekeys are more discriminative than synsets and this favors the disambiguation.

[5]*senseval2* contains 2,282 instances, of which the system would answer incorrectly 135 instances if the MFS fallback strategy is not used, hence dropping 0.06 in performance.

Model	senseval2 sensekey F_1	senseval2 synset F_1	semeval2013 sensekey F_1	semeval2013 synset F_1
Our LSTM (T: SemCor)	**0.726**	**0.720**	0.661	0.647
Our LSTM (T: OMSTI)	0.688	0.675	0.655	0.651
Our LSTM (T: SemCor+OMSTI)	0.703	0.699	**0.667**	**0.656**

Table 5: Comparison of F_1 scores of our implementation using either synset or sensekey level to represent meanings.



(a) Performance vs. unannotated corpus size (b) Performance vs. number of parameters

Figure 2: The effect of (a) the size of unannotated corpus and (b) the number of parameters on WSD performance. Number of parameters includes the weights of the hidden layer, the weights of the projection layer, and the input and output embeddings. Notice that the horizontal axis is in log scale.

Impact of unannotated data and model size. Since unannotated data is abundant, it is tempting to use more and more data to train language models, hoping that better word embeddings would translate into improved WSD performance. The fact that Yuan et al. (2016) used a 100-billion-token corpus only reinforces this intuition. We empirically evaluate the effectiveness of unlabeled data by varying the size of the corpus used to train LSTM models and measure the corresponding WSD performance. More in particular, the size of the training data was set at 1%, 10%, 25%, and 100% of the GigaWord corpus (which contains 1.8×10^7, 1.8×10^8, 4.5×10^8 and 1.8×10^9 words, respectively).

Figure 2a shows the effect of unannotated data volume on WSD performance. The data points at 100 billion (10^{11}) tokens correspond to Yuan et al. (2016)'s reported results. As might be expected, a bigger corpus leads to more meaningful context vectors and therefore higher performance on WSD. However, the amount of data needed for 1% of improvement in F_1 grows exponentially fast (notice that the horizontal axis is in log scale). Extrapolating from this graph, to get a performance of 0.8 F_1 by adding more unannotated data, one would need a corpus of 10^{12} tokens. This observation also applies to the balance of the sense assignment. Using only 25% of the unannotated data already yields a recall of 35% on the less-frequent senses.

In addition, one might expect to push the performance further by increasing the capacity of the LSTM models. To evaluate this possibility, we performed an experiment in which we varied the sizes of LSTM models trained on 100% of the GigaWord corpus and evaluated against *senseval2* and *semeval2013*, respectively. Figure 2b suggests that it is possible but one would need exponentially bigger models.

Finally, Reimers and Gurevych (2017) have showed that it is crucial to report the distribution of test scores instead of only one score as this practice might lead to wrong conclusions. As pointed out at the beginning of Section 5, our biggest models take months to train, making training multiple versions of them impractical. However, we trained our smallest model ($h = 100, p = 10$) ten times and our second smallest model ($h = 256, p = 64$) five times and observed that as the number of parameters increased, the standard deviation of F_1 decreased from 0.008 to 0.003. We, therefore, believe random fluctuation does not affect the interpretation of the results.

6 Conclusions

This paper reports the results of a reproduction study of the model proposed by Yuan et al. (2016) and an additional analysis to gain a deeper understanding of the impact of various factors on its performance.

A number of interesting conclusions can be drawn from our results. First, we observed that we do not need a very large unannotated dataset to achieve state-of-the-art all-words WSD performance since we used the Gigaword corpus, which is two orders of magnitude smaller than Yuan et al. (2016)'s proprietary corpus, and got similar performance on *senseval2* and *semeval2013*. A more detailed analysis hints that adding more unannotated data and increasing model capacity are subject to diminishing returns. Moreover, we observed that this approach has a more balanced sense assignment than other techniques, as shown by the relatively good performance on less-frequent-sense instances. In addition, we identified that the limited sense coverage in annotated dataset places a potentially upper bound for the overall performance.

The code with detailed replication instructions is available at: `https://github.com/cltl/wsd-dynamic-sense-vector` and the trained models at: `https://figshare.com/articles/A_Deep_Dive_into_Word_Sense_Disambiguation_with_LSTM/6352964`.

Acknowledgements

The research for this paper was supported by the Netherlands Organisation for Scientific Research (NWO) via the Spinoza-prize Vossen projects (SPI 30-673, 2014-2019). We thank the support of the NWO project scilens (`https://projects.cwi.nl/scilens`) for providing the hardware to run some of the experiments. Experiments were also carried out on the Dutch national e-infrastructure with the support of SURF Cooperative. We would like to thank our friends and colleagues Antske Fokkens, Emiel van Miltenburg, Chantal van Son, Pia Sommerauer, and Roxane Segers for many useful comments and discussions.

References

Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Rajat Monga, Sherry Moore, Derek Murray, Jon Shlens, Benoit Steiner, Ilya Sutskever, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Oriol Vinyals, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467v2*.

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2006. Relation extraction using label propagation based semi-supervised learning. In *ACL 2016*, pages 129–136.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack Long Short-Term Memory. In *ACL 2015*, pages 334–343.

Andreas Eisele and Yu Chen. 2010. MultiUN: A multilingual corpus from united nation documents. In *LREC 2010*.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May.

Winthrop Nelson Francis and Henry Kucera. 1979. Brown corpus manual. *Brown University*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. *ACL 2017*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *ACL 2016*, pages 897–907. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL-IJCNLP 2015*, pages 1–10.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. Context2vec: Learning generic context embedding with bidirectional LSTM. In *CoNLL*, pages 51–61.

George Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Human language technology: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *TACL 2014*, 2:231–244.

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: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA, June. 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 *ACL 2005*, pages 395–402, Stroudsburg, PA, USA. Association for Computational Linguistics.

Martha Palmer, Christiane Fellbaum, Scott Cotton, Lauren Delfs, and Hoa Trang Dang. 2001. English tasks: All-words and verb lexical sample. In *Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems (SENSEVAL-2)*, pages 21–24, Toulouse, France, July. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Marten Postma, Ruben Izquierdo Bevia, and Piek Vossen. 2016. More is not always better: balancing sense distributions for all-words word sense disambiguation. In *COLING 2016: Technical Papers*, pages 3496–3506.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017a. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *EACL 2017*, pages 99–110. Association for Computational Linguistics.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017b. Neural sequence learning models for word sense disambiguation. In *EMNLP 2017*, pages 1156–1167. Association for Computational Linguistics.

Sujith Ravi and Qiming Diao. 2015. Large Scale Distributed Semi-Supervised Learning Using Streaming Approximation. *arXiv:1512.01752 [cs]*, 51.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *EMNLP 2017*, pages 338–348.

Sascha Rothe and Hinrich Schütze. 2017. Autoextend: Combining word embeddings with semantic resources. *Computational Linguistics*, 43(3):593–617.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kaveh Taghipour and Hwee Tou Ng. 2015. One million sense-tagged instances for word sense disambiguation and induction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 338–344. Association for Computational Linguistics.

Rocco Tripodi and Marcello Pelillo. 2017. A game-theoretic approach to word sense disambiguation. *Computational Linguistics*, 43(1):31–70.

Dirk Weissenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. 2015. Multi-objective optimization for the joint disambiguation of nouns and named entities. In *ACL 2015*, pages 596–605. Association for Computational Linguistics.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In *COLING 2016: Technical Papers*, pages 1374–1385. The COLING 2016 Organizing Committee.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A wide-coverage word sense disambiguation system for free text. In *ACL 2010: System Demonstrations*, pages 78–83. Association for Computational Linguistics.

Guo-Dong Zhou. 2011. Learning noun phrase anaphoricity in coreference resolution via label propagation. *Journal of Computer Science and Technology*, 26(1):34–44.

Xiaojin Zhu and Z Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. *CMU CALD tech report*, (CMU-CALD-02-107).