# Word Sense Disambiguation with Recurrent Neural Networks

**Alexander Popov**
Linguistic Modelling Department
IICT-BAS
alex.popov@bultreebank.org

## Abstract

This paper presents a neural network architecture for word sense disambiguation (WSD). The architecture employs recurrent neural layers and more specifically LSTM cells, in order to capture information about word order and to easily incorporate distributed word representations (embeddings) as features, without having to use a fixed window of text. The paper demonstrates that the architecture is able to compete with the most successful supervised systems for WSD and that there is an abundance of possible improvements to take it to the current state of the art. In addition, it explores briefly the potential of combining different types of embeddings as input features; it also discusses possible ways for generating "artificial corpora" from knowledge bases – for the purpose of producing training data and in relation to possible applications of embedding lemmas and word senses in the same space.

## 1 Introduction

The task of word sense disambiguation (WSD) has been defined as follows: "the ability to computationally determine which sense of a word is activated by its use in a particular context." (Navigli, 2009) Work on this problem has a long history and it is generally recognized as one of the field's most difficult tasks, for a variety of reasons ranging from data sparseness and the difficulty in constructing good lexicons to the inherent complexity of the task itself. Several broad families of methods have been proposed: unsupervised, supervised and knowledge-based methods (following the classification in Navigli (2009); note that sometimes these approaches might overlap, e.g. semi-supervised ones have also led to some promising results (Yuan et al., 2016b)). Among these, supervised systems typically achieve the best results, as is comprehensively shown in Raganato et. al. (2017). Consequently, the challenge to improving WSD is threefold: increasing the amount of available training data, extracting richer input features for the learning algorithms, improving the learning algorithms themselves.

The task of generating more training data is notoriously difficult and expensive. In contrast to other similar classification tasks, such as part-of-speech tagging, the variety of possible tags and word-tag pairs in WSD is enormous (consider the following comparison: the popular Penn Treebank POS tagset contains 58 tags, whereas WordNet 3.1, one of the most popular word sense inventories, contains over 117,000 synsets). This raises a number of issues, such as data sparseness (some word senses are underrepresented in the data, or not represented at all) and low inter-annotator agreement (word senses are often too fine-grained, which results in contradictory selections done by the human annotators; low inter-annotator agreement (IAA) is a sign of a theoretical ceiling to what an automated system trained and tested on such data could achieve; e.g., Navigli (2009) reports IAA for WordNet-style senses to be between 67 and 80%).

The two remaining issues are more amenable to improvement, especially with recent developments in the use of neural networks for NLP tasks. This work will examine the usage of neural architectures in the context of WSD, as well as the usage of embeddings as input features, thus tackling both problems (feature extraction and algorithm improvement). Its contribution is twofold: demonstrating that recurrent neural networks (RNNs) are suitable for solving the all-words lexical disambiguation task and exploring new direc-


*Proceedings of the Student Research Workshop associated with RANLP 2017*, pages 25–34,
Varna, Bulgaria, 4-6 September 2017.
http://doi.org/10.26615/issn.1314-9156.2017_004

tions for generating and using embeddings, with respect to the WSD task. It also briefly outlines an option for generating "artificial" data that can be used for training word and sense representations, or for training the WSD algorithms themselves.

The following section provides references to related work in the field that motivates the present study, then Section 3 describes the neural architecture. Section 4 describes the data used and the experimental setup. Section 5 gives the empirical results. Section 6 concludes the paper and outlines possible further work.

## 2 Related Work

### 2.1 Neural Network Language Models and Word Representations

Neural network language models have driven a wave of improvements in NLP tasks in recent years, due to a great extent to the widespread use of word representations, also known as "word embeddings". Word embeddings are real-valued vector representations of words in a lower-dimensional space (as opposed to, for instance, "one-hot" representations, whereby the dimensionality is equal to the lexicon size). The training of embeddings can be accomplished in different ways (e.g. using a feedforward neural network as in the pioneering work of Bengio et. al. (2003) or using a convolutional one such as in Collobert and Weston (2008)).

One of the most significant contributions to the field has been Mikolov et. al. (2013), which proposes fast and efficient methods for the training of embeddings on large corpora – using a simple log-linear feedforward network. The goal of the network is to predict a hidden section of the input text based on a visible context (two variants are introduced – CBOW and Skipgram). The intermediate matrix that performs the embedding in the lower-dimensional space is what is of greatest interest, as it provides the mapping from naive one-hot representation to distributed representation of meaning, which is also much more compact. It is demonstrable through simple arithmetic operations on the representations that they are able to encode meaningful distinctions along different semantic dimensions (e.g. number, sex, functional roles, etc.). Subsequently, other approaches to obtaining embeddings from large natural language corpora have been proposed (e.g., see Levy and Goldberg (2014) for dependency-based embeddings, Pennington et. al. (2014) for embeddings that reflect co-occurrence statistics in a large corpus).

### 2.2 Word Embeddings as Features to Supervised Systems

Distributed word representations can be very strong features for supervised models and have been successfully used in WSD systems. Taghipour and Zhong (2015) describe a way to incorporate word embeddings in a popular supervised system – IMS (Zhong and Ng, 2010); the addition of the embeddings leads to a significant improvement in accuracy when the system is tested on several different datasets. The traditional features used by IMS are binary ones that indicate information about the window-bounded textual context of the words to be disambiguated (such as POS, lemma, etc); word embeddings are added as real-valued features which do not fit well with the binary ones and therefore the authors use a scaling method from Turian et. al. (2010). A later study (Iacobacci et al., 2016) surveys different methods for integrating embeddings in the IMS system, such as concatenating or averaging the vectors for the surrounding words or weighing the vectors via *fractional* or *exponential decay*; the exponential decay method gives very good improvements, especially on the Senseval (SE) lexical sample tasks (SE 2, 3 & 7).

### 2.3 Recurrent Neural Networks for Word Sense Disambiguation

The IMS system discussed above uses an SVM algorithm to perform the disambiguation. A sensible question is – are neural networks a good alternative for WSD? This overview will focus on several neural approaches proposed recently.

One important development has been the adoption of recurrent neural networks (RNNs) as a viable tool for modeling language. RNNs are similar to deep feedforward networks, with the major difference that the hidden (context) layers have cyclic connections to themselves, which allows them to maintain a dynamic memory of their previous states, as the latter change in time. This ability to keep a memory trace of past "events" at theoretically arbitrary distances from the present is an obvious advantage over algorithms that collect information from a fixed window around the target word. Especially in the case of more complex tasks such as WSD, vital information is often found at the other end of the time series (i.e. the

sentence; sometimes the disambiguation of a word might require going back even before a sentence boundary, and on rare occasions it might even depend on looking forward and beyond the current sentence).

For a long time RNNs were considered difficult to train, as their memory capabilities are often thwarted in practice by the phenomena known collectively as the *exploding/vanishing gradients problem* – the fact that with long time series the backpropagated error gradients often grow too large or too small. While the exploding gradients part can be solved trivially by capping their values, a more elaborate solution was needed for the vanishing part. *Long Short Term Memory* (LSTM) cells (Hochreiter and Schmidhuber, 1997) were deliberately designed to be able to selectively forget (parts of) old states and pay attention to new inputs (a good introduction to LSTMs is Graves (2012); another similar and newer development are *Gated Recurrent Units* (Cho et al., 2014)).

A further and simple enhancement to such an architecture is making it *bidirectional* (Graves and Schmidhuber, 2005), i.e. the input sequence is fed into two recurrent context layers – one running from the beginning to the end of the sequence and the other one running in reverse. Their outputs are concatenated and thus encompass *forward* as well as *backward*-looking context. Bidirectional LSTMs (Bi-LSTMs) have been successfully applied to a number of sequence-to-sequence tasks in NLP, such as part-of-speech tagging, chunking, named entity recognition, dependency parsing ((Wang et al., 2015a), (Wang et al., 2015b), (Huang et al., 2015), (Wang and Chang, 2016)).

RNNs have been used in several ways for WSD. One such work is Kågebäck and Salomonsson (2016), which uses a Bi-LSTM to solve the *lexical sample tasks* of Senseval-2 (Kilgarriff, 2001) and Senseval-3 (Mihalcea et al., 2004) – that is, the model is disambiguating one word per sentence only. To that purpose, the output of the Bi-LSTM at the target word position is fed upstream for the classification part; it takes into consideration both the left and the right contexts. It is reshaped through a lemma-specific hidden layer, so that classification between the possible senses for the lemma can be carried out. In this way the model shares the Bi-LSTM parameters across words and is updated globally with every training case, but is parametrized for specific lemmas. The model is on par (or slightly better) with state-of-the-art systems, but uses no other features apart from the word embeddings.

Another approach to WSD that also uses RNNs is via calculating vector similarities. The goal of such models is to calculate a distributed representation both of the target sense and of the context within which it appears. Upon doing that, some similarity measure, such as cosine distance, can be used to determine which of the possible word senses for the target word is closest to the context representation. Such models are typically not supervised, because they do not update their parameters against training data annotated with word sense information; however, they do rely heavily on a pre-training procedure that enables the network to represent contexts as embeddings. RNNs are useful in this case, because they can capture well syntactic information, as opposed to bag-of-word approaches. The embeddings for the different word senses are obtained by running the example sentences for them (e.g. those supplied in their WordNet entries) through the RNN. Naturally, the more data that is available, the better sense representations can be built. Examples of such models can be found in Yuan et. al. (2016a), Melamud et. al. (2016) and Le and Mikolov (2014).

The current work explores both RNN-based approaches to WSD outlined above, with respect to the all-words lexical task.

### 2.4 Relational Knowledge and Generation of Embeddings

This subsection of the literature review will take into account a last constellation of ideas that are also important to the currently presented work. As was already discussed, word embeddings are very effective as features to supervised models. Word embeddings are typically generated by training models on large amounts of unlabeled natural language. But other sources of information, such as, for instance, dependency paths, can also yield useful training signals.

One innovative approach to the embedding of words is Goikoetxea et. al. (2015). That work uses a *knowledge base* (more specifically WordNet) to generate an artificial (or *pseudo*) corpus, which is then used to generate embeddings. The pseudo-corpus is created by utilizing WordNet's relational structure (including relations such as hypernymy, synonymy, antonymy, derivation, etc.) – more specifically by traversing the semantic network and emitting a word sense identifier for each

node in the graph that is visited. The traversal is accomplished via the Pagerank algorithm (Page et al., 1999), as implemented in the UKB tool (Agirre and Soroa, 2009). Several million "random walks" along the graph are produced in this way and each node along these sequences is replaced with a representative lemma (taken from the WordNet entry for the sense). The pseudo-corpus is then fed in the Word2Vec tool, which embeds the lemmas in a lower-dimensional space. The embeddings are tested against a set of word embeddings taken from Mikolov et. al. (2013) – on popular similarity and relatedness datasets. The experiments show them to be very competitive and even more effective in some cases.

Simov et. al. (2017) builds on this work by enriching the knowledge graph with different types of relations, such as syntagmatic relations extracted from the WordNet glosses and inference over the hypernymy relations already present there. Some of those expanded graphs yield embeddings that further improve the performance on the similarity and relatedness tasks. The results suggest that knowledge-based approaches to generating data are viable alternatives to working with huge amounts of text (often amounting to hundreds of billions of tokens). Goikoetxea et. al. (2016) in turn demonstrates that word representations learned in this way can be viewed not simply as an alternative, but as a complement to embeddings learned from distributional knowledge. Even simple methods of combining distributional and relational-based embeddings, such as vector concatenation, are show to improve accuracy for the similarity and relatedness tasks. This line of work is also valuable due to the possibilities it opens for training genuine *sense embeddings*.

Regarding the generation of sense embeddings, it is worth mentioning a few more attempts at obtaining such vector representations that go beyond averaging the word embeddings for representative sentences. Chen et. al. (2014) obtain their sense embeddings from the sense glosses, but they filter out some of the words in the definition (functional words and such that fail a cosine similarity comparison with the lemma in question). Johansson and Piña (2015b; 2015a) present a method of embedding the senses together with the lemmas for the WordNet entries which minimizes a metric of *neighborhood* calculated on the basis of the relational structure in WordNet, with the lemma embeddings being represented as convex combinations of the different possible senses. Rothe and Schütze (2015) derive sense and lexeme embeddings from word embeddings through autoencoders; their solution does not require any additional training resources beyond the word embeddings. The availability of word/lemma embeddings in the same space with sense embeddings opens up exciting possibilities for doing WSD, some of which are explored in this work.

## 3 Neural Network Architecture for WSD

### 3.1 Network Implementation

The architecture implemented for this study employs a Bi-LSTM context layer, similar to Kågebäck and Salomonsson (2016). Unlike that model, however, it is designed to perform disambiguation for *all* open-class words in a single context. This means that an input sequence is fed one token at a time into the Bi-LSTM and for each target word the output of the context layer is passed upstream in order to eventually feed a classification layer. That is, this architecture's context layer produces a number of outputs per sentence, much like in other sequence-to-sequence tasks such as POS tagging, as opposed to just one representation for a specific word or for the whole context.

The architecture is represented graphically in Figure 1. The input sequence of words are preprocessed from string to integer format, where each integer corresponds to a position in an embedding matrix. A parameter setting allows some of the words in the training sequences to be dropped, which should hypothetically reduce overfitting (this feature is a modified version of *dropword* from Kågebäck and Salomonsson (2016); however, instead of replacing randomly selected words with a $\langle dropped \rangle$ tag, words here are directly deleted from the training input). The integer inputs are then fed into an embedding layer that selects the corresponding vectors (this layer is also trainable, i.e. the embeddings continue to adapt as the WSD model is being trained and could be saved and reused in other tasks that would benefit from such adaptation). The network is parametrized to be able to access two parallel embedding layers, so that embeddings for the same words but from separate sources can be combined at this stage (via simple concatenation).

Consequently, the input vectors are fed into a Bi-LSTM layer that runs them sequentially through itself, analyzing the time series simul-

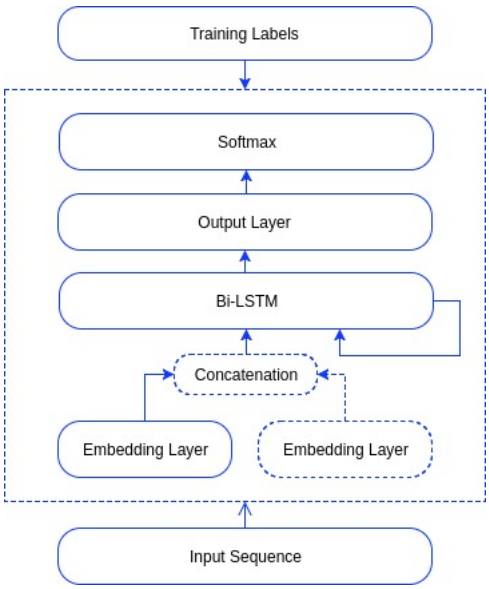


Figure 1: Recurrent neural network for word sense disambiguation: The dotted lines mean that a component or a connection is optional (in the case of concatenating embeddings from two different sources).

taneously from-beginning-to-end and from-end-to-beginning. Dropout (Srivastava et al., 2014) can be added to both the input and the output of the forward and backward cells of the Bi-LSTM; the network can have an arbitrary number of Bi-LSTM layers stacked on top of each other. The outputs of the context layer are filtered out in order to extract just the ones corresponding to target words. These are then put through a linear hidden layer that transforms each input (dimension = 2 * number of neurons in each LSTM cell) into a large vector with one position per every word sense that corresponds to a lemma seen in the training data. There is a softmax layer on top, which creates a probability distribution over the word sense lexicon. Finally, cross entropy loss is calculated between this vector and the gold label and the mean value for the batch is passed to the optimizer. The implementation is written within the Tensorflow framework (Abadi et al., 2016).

### 3.2 WSD Features

The model uses no hand-crafted features during the training process, all the information comes from the embeddings that are loaded as the first layer of the network. As principal features, the GloVe embeddings were used (Pennington et al., 2014), more specifically the set trained on Wikipedia and Gigaword (6 billion tokens text, vocabulary of 400K words, uncased, vector dimensionality is 300). It remains as a future task to test the other available options, some of which have been trained on far larger corpora, as well as other corpora-based word representation methods altogether.

The option for a second embedding layer, parallel to the default one, was added so that the ideas presented in subsection 2.4 could be tested in such a setup. An extension to WordNet was selected on the basis of Simov et. al. (2017) – the so-called "WN30WN30glConOne", one of the best performing graphs on the similarity and relatedness tasks. The new relations in it are constructed on the basis of co-occurring word sense annotations in the WordNet glosses, taken from eXtended WordNet (Mihalcea and Moldovan, 2001); for a more detailed description, see Simov et. al (2016). The UKB[1] tool was used with the reported settings to replicate this pseudo-corpus (as described in subsection 2.4, producing 500 million random walks along the graph structure. Then Word2Vec[2] was used to create lemma embeddings based on the artificial corpus, again using the settings indicated in the referenced article:

- Size of the embeddings = 300
- Number of training iterations = 7
- Number of negative examples = 5
- Window size = 15
- Threshold for occurrence of words = 1e-7
- Algorithm = Skipgram

This method provides representations only for lemmas and only for open-class words. Therefore a lot of the input words (mostly functional ones) do not have matching vectors. And because embeddings are created only with respect to lemmatized input, morphological information is largely disregarded. However, if such pseudo-corpora truly generate information that can complement what is learned from large natural language corpora, there is a plausible hypothesis that such an embedding could improve the accuracy of a WSD system. The combination method (concatenation) was chosen because it is easy to implement and because of the evidence provided in Goikoetxea et. al. (2016) that it gives adequate results compared to more complex ones.

[1] http://ixa2.si.ehu.es/ukb/
[2] https://code.google.com/archive/p/word2vec/

### 3.3 Artificial Corpora with Mixed Lemmas and Word Senses

By employing the procedure described in subsection 3.2, one can generate a pseudo-corpus that is a mix of word sense IDs and representative lemmas (this can be done easily via one of the parameters provided by the UKB tool). Based on the previous discussion of the literature, such a corpus can serve for at least two purposes:

1. In order to embed lemmas and word senses simultaneously in the same space. These representations can be used in approaches such as those discussed in subsection 2.3 – for calculating context embeddings and comparing them with sense embeddings.

2. In order to generate artificial training data for supervised WSD systems. The fact that such corpora can be used as training data for the generation of meaningful embeddings, suggest that they could have some worth as input to WSD models as well.

Naturally, such a corpus would be very noisy, especially if it is to be used as training data for WSD. The proposed method is quite naive in that lemmas are picked randomly to substitute the sense IDs, but more sophisticated strategies can be devised as well. For instance, the glosses for the WordNet synsets can be added automatically next to the IDs generated by UKB. In this way more meaningful linguistic data will be intermixed with the (semi-) random walks performed by the Pagerank algorithm. This would however require a rethinking of the Word2Vec parameters used, as distances between individual sense IDs are likely to increase substantially (the gloss annotations in Mihalcea et. al. (2001) can be used to ameliorate this problem). This approach will allow for the learning of embeddings for functional words as well; otherwise there would be no information about them that could be passed to the RNN.

Here, a tentative modification to the already described architecture is proposed that makes use of such an artificial corpus and the lemma/sense representations generated from it. This work is in its earliest stages, but could potentially connect in a meaningful way to the rest of the ideas discussed here. The modification is simple – instead of mapping the RNN outputs to a huge lexicon-sized vector, the final hidden layer produces a vector the size of the sense embeddings. No softmax is applied thereafter, but the sense embedding for the gold label word sense is compared with the one produced by the network (different cost functions can be used to that purpose, such as cosine distance, etc; a naive one is used here – least squares). This training signal is used to optimize the parameters of the network, so that it can learn how to calculate context embeddings for specific target words in a more informed manner. The disambiguation is done by picking the closest sense, via cosine similarity.

Such a method has the advantage that it optimizes on all of the semantic dimensions encoded by the sense embeddings – i.e. with each training case, the network should be learning important information about the whole semantic space, not just about a single label that it aims to pick amongst the rest in the lexicon. To test this solution, a mixed corpus of size 200 million random walks was produced, where the probability of emitting a sense ID or a lemma is evenly split. The same Word2Vec settings were used as the ones reported above for the lemma embeddings.

## 4 Data and Experimental Setup

### 4.1 Data and Evaluation Procedure

For the purpose of this work, the data referenced in the Unified Evaluation Framework (UEF) by Raganato et. al. (2017) was used, in order to be able to compare the experimental results with the ones reported there[3]. Consequently, SemCor (Miller et al., 1994) was used for training, as that is one of the datasets against which the supervised systems within the UEF have been trained. The all-words lexical task in Senseval-2 (Kilgarriff, 2001) was used for testing. The WordNet synset IDs provided in the gold key were mapped to the WordNet 3.0 lexicon, also used to map lemmas to possible synsets.

Since the Senseval-2 dataset provides information about the lemmatized form and the POS tag of the words (which are converted to features by some of the supervised systems; the Stanford CoreNLP toolkit[4] is used for the POS tagging), this information has been used as a filter at the evaluation step (i.e. outside of the neural network itself). First of all, for all lemmas that are used as

[3]The data, results and descriptions in the UEF are available online at http://lcl.uniroma1.it/wsdeval/

[4]https://stanfordnlp.github.io/CoreNLP/

target words and have only one associated WordNet sense, that synset is chosen automatically during evaluation. Additionally, when evaluating a particular word, the WN synsets associated with it that bear different POS tags from the one indicated in the gold data are disregarded as possible options. Whenever the gold data indicates that more than one sense is a correct choice in a particular case, the system is evaluated to be correct if it selects one of them as most probable. And finally, whenever a lemma is encountered that has not been seen in the training data, the system falls back to the most-frequent-sense heuristic, using the frequency information in WordNet.

### 4.2 Experimental Setup

Table 1 below gives the parameters with which the neural network achieved the highest results on the evaluation data.

| Parameter | Value |
|---|---|
| Embedding size | 300 |
| Word embeddings | GloVe |
| Lemma embeddings | WN30WN30glConOne |
| Bi-LSTM hidden units | 2 * 200 |
| Bi-LSTM layers | 1 |
| Dropout | 20% |
| Dropword | 0% |
| Optimizer | SGD |
| Learning rate | 0.2 |
| Initialization of LSTMs | random uniform [-1;1] |
| Training batch size | 100 |
| Training epochs | 100K |

Table 1: Parameters for the highest result obtained with the neural network.

The presence of both the GloVe vectors and the lemma embeddings in the table means that each embedding layer feeds a 300-position vector as input per word, not that the post-concatenation vector has 300 positions (that is, the final input has a dimensionality of 600). Changing the size of the LSTMs below or above 200 units has negative results on accuracy, as does adding more than one Bi-LSTM layer. Dropout is typically set to 50%, but in this case a setting of 20% produced the best results in the explored range of [0:50]. Dropword, at least the way it is applied here, does not seem to improve the results, although more experimental work is needed in order to try out a wider range of values. It remains on the agenda to test more sophisticated optimization algorithms than Stochastic Gradient Descent, as well as to integrate a learning rate decay, momentum and additional regularization techniques.

With regards to the solution outlined at the end of subsection 3.3, the parameters of the network are mostly the same. The only difference is that just one embedding layer of size 300 is used (since only the lemma embeddings are utilized) and that the number of the Bi-LSTM hidden units is 2*400 instead of 2*200.

## 5 Results and Discussion

Before presenting a comparison of the experimental results with other systems, Table 2 provides some select information about how different parameters influence the accuracy of the network. The top entry gives the best score that has been obtained (with the parameters in the previous table); the second entry gives the highest accuracy score obtained without dropout regularization and the third one gives the highest score obtained without dropout and without the lemma embeddings. The effect of dropout is not very big, but the addition of the lemma embeddings turns out to provide a powerful boost.

| Combination | Accuracy |
|---|---|
| Best | **70.11%** |
| No dropout | 69.98% |
| No lemma embeddings | 68.97% |

Table 2: Comparison between the best recorded setting and two other combinations of parameters.

Table 3 puts the results of this particular work in the context of the best-scoring supervised systems tested on the same dataset. The numbers are taken from Raganato et. al. (2017)[5], which gives a comparison of several such systems trained and tested on the same datasets. The results for choosing a random and the most frequent sense (MFS) are also included (the latter is a very hard baseline to beat in general, especially when good statistics are available), as well as the best results reported there for a knowledge-based system (a specific configuration of the UKB tool called UKB-g*).

[5]Note that the IMS results provided there are higher from the previous ones in Iacobacci at. al. (2016) and Zhong and Ng (2010). Thus, the 2016 results for IMS + Word2Vec (trained on SemCor) are also included ("IMS-2016"), as well as the original results from 2010 ("IMS-2010").

| System | Accuracy |
|---|---|
| **IMS-s+emb** | **72.2%** |
| Context2Vec | 71.8% |
| **This work** | **70.11%** |
| UKB-g* | 68.8% |
| IMS-2010 | 68.2% |
| MFS | 65.6% |
| IMS-2016 | 63.4% |
| **This work - vector similarity** | **61.44%** |
| Random | 41.93% |

Table 3: Comparison of this work with other systems trained on SemCor and tested on the Senseval-2 all-words lexical task. *IMS-s+emb*, *Context2Vec*, *UKB-g** and *MFS* are reported in Raganato et. al. (2017), *Random* is the performance of the current system prior to any learning.

The results show that the neural network model presented here is not lagging too far behind some of the highest reported scores on this particular dataset. Further optimizations of the network parameters, as well as more powerful input features, could bring it to par with the state of the art. The neural network is comfortably ahead of the MFS result and it also scores higher than the best knowledge-based candidate. Especially encouraging is the observation that the lemma embeddings produced via an "artificial" corpus significantly improve the score, meaning that there is still more space for enriching the input features.

Finally, a comment regarding the proposed solution that employs a vector similarity metric (introduced at the end of subsection 3.3). The result shown here is significantly lower than that of the other systems. Nevertheless, there are reasons to think that this line of research is worth pursuing. The proposed architecture does learn meaningful information, because it fares much better than a random choice heuristic. It continues to improve its accuracy even around the end of the 100K training epochs and it achieves this result with an almost unoptimized parameter configuration and with a set of lemma/sense-embeddings that is somewhat naive (it contains no information about functional words or morphologically-coded grammatical information).

## 6 Conclusion and Future Work

This work has presented a supervised architecture for WSD using recurrent neural networks, tested on the all-words lexical task. The model is inspired by the very successful recent applications of LSTM cells to NLP problems. The article also explores the utility of combining word embeddings learned from a large corpus of text with lemma embeddings learned from an "artificially generated" corpus based on a knowledge resource (WordNet). A comparison with the best-scoring systems on a popular evaluation dataset shows that the neural network is well-positioned with respect to them. The fact that the addition of the lemma embeddings from the pseudo-corpus improves significantly the results, signals that they could be further boosted by exploring different feature spaces and combinations of them.

There are a number of changes and additions that could be undertaken to improve the WSD algorithm. Further parameter optimization is naturally among them, as is the evaluation of other sets of word representations and the generation of improved "artificial corpora". Another improvement that could boost accuracy scores and for which RNNs are naturally suited, is allowing the model to store a representation for the previously seen sentences, i.e. to disambiguate words using whole texts as context, rather than just separate sentences. While this would not matter in tasks such as POS tagging and syntactic parsing, it is very important with regards to WSD.

And finally, a network could be trained to jointly solve two related tasks: the WSD one and that of adapting context embeddings to word sense embeddings (as discussed in 3.3). Since the similarity task is optimizing the RNN with respect to all semantic dimensions of the embedding space, the classification pathway should be able to directly benefit from that. This provides further motivation for improvements in the generation of less naive "mixed" corpora. It also remains to be tested whether such pseudo-corpora can be successfully used as training data for the supervised WSD systems themselves (as complements to expensive corpora such as SemCor).

## Acknowledgements

This research has received partial support by the grant 02/12 — *Deep Models of Semantic Knowledge (DemoSem)*, funded by the Bulgarian National Science Fund in 2017–2019. I am grateful to the anonymous reviewers for their remarks, comments, and suggestions, and to my supervisor, Kiril Simov, for his support and guidance.

## References


Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

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

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

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

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 160–167.

Josu Goikoetxea, Eneko Agirre, and Aitor Soroa. 2016. Single or multiple? combining word representations independently learned from text and wordnet. In *AAAI*. pages 2608–2614.

Josu Goikoetxea, Aitor Soroa, Eneko Agirre, and Basque Country Donostia. 2015. Random walks and neural network language models on knowledge bases. In *HLT-NAACL*. pages 1434–1439.

Alex Graves. 2012. Supervised sequence labelling. *Supervised sequence labelling with recurrent neural networks* pages 5–13.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5):602–610.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *ACL (1)*.

Richard Johansson and Luis Nieto Pina. 2015a. Combining relational and distributional knowledge for word sense disambiguation. In *Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA 2015, May 11-13, 2015, Vilnius, Lithuania*. Linköping University Electronic Press, 109, pages 69–78.

Richard Johansson and Luis Nieto Pina. 2015b. Embedding a semantic network in a word space. In *HLT-NAACL*. pages 1428–1433.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. *arXiv preprint arXiv:1606.03568* .

Adam Kilgarriff. 2001. English lexical sample task description. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*. Association for Computational Linguistics, pages 17–20.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1188–1196.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *ACL (2)*. pages 302–308.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *CoNLL*. pages 51–61.

Rada Mihalcea, Timothy Chklovsky, and Adam Kilgarriff. 2004. Itri-04-09 the senseval-3 english lexical sample task. *Information Technology* 25:28.

Rada Mihalcea and Dan I. Moldovan. 2001. extended wordnet: progress report. In *in Proceedings of NAACL Workshop on WordNet and Other Lexical Resources*. pages 95–100.

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

George A Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G Thomas. 1994. Using a semantic concordance for sense identification. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pages 240–243.

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

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

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

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *arXiv preprint arXiv:1507.01127* .

Kiril Simov, Petya Osenova, and Alexander Popov. 2016. Using context information for knowledge-based word sense disambiguation. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*. Springer, pages 130–139.

Kiril Simov, Petya Osenova, and Alexander Popov. 2017. Comparison of word embeddings from different knowledge graphs. In *International Conference on Language, Data and Knowledge*. Springer, pages 213–221.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *HLT-NAACL*. pages 314–323.

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*. Association for Computational Linguistics, pages 384–394.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015a. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168* .

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015b. A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *arXiv preprint arXiv:1511.00215* .

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *ACL (1)*.

Dayu Yuan, Ryan Doherty, Julian Richardson, Colin Evans, and Eric Altendorf. 2016a. Word sense disambiguation with neural language models. *CoRR* .

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016b. Semi-supervised word sense disambiguation with neural models. *arXiv preprint arXiv:1603.07012* .

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