Abstract

In this paper we present a model that leverages a bidirectional long short-term memory network to learn word sense disambiguation directly from data. The approach is end-to-end trainable and makes effective use of word order. Further, to improve the robustness of the model we introduce dropword, a regularization technique that randomly removes words from the text. The model is evaluated on two standard datasets and achieves state-of-the-art results on both datasets, using identical hyperparameter settings.

1 Introduction

Words are in general ambiguous and can have several related or unrelated meanings depending on context. For instance, the word rock can refer to both a stone and a music genre, but in the sentence "Without the guitar, there would be no rock music" the sense is no longer ambiguous. From this example it is easy to see that the context surrounding the word is what defines the sense. However, it may not be so obvious that this is a difficult task. To see this, consider instead the sentence "Hard rock crushes heavy metal" where all words in themselves seems to indicate stone but together they actually define the word token as music. Hence, to do word sense disambiguation (WSD) well we need to go beyond bag of words and into the territory of sequence modeling.

Improved WSD would be beneficial to many of the big natural language processing (NLP) problems, e.g. machine translation, information Retrieval, information Extraction (Vickrey et al., 2005; Navigli, 2009), and basic NLP tasks like sense aware word representations (Neelakantan et al., 2015; Nieto Piña and Johansson, 2015; Bovi et al., 2015). However, for word representations the benefit goes two ways which was shown by Taghipour and Ng (2015) who used word embeddings to improve on WSD, and later by Johansson and Pina (2015) who combined word embeddings with a knowledge graph approach. Though much progress has been done in the area, most current WSD systems suffer one of two problems. They either disregard the order of words or they rely on complicated hand crafted features. The first problem will lead to systems with low precision since many different contexts, potentially corresponding to different senses, will consist of the same bag of words. The second problem usually lead to high precision but will inevitably lack in coverage and therefore produce a low recall system. We aim to mitigate these problems by (1) modeling the sequence of words surrounding the target word instead of a bag, and (2) refrain from using any hand crafted features but let the model learn the features directly from the text during training.

The main contributions of this work include:

- An end-to-end trainable WSD system that achieves state-of-the-art result on two standard datasets without any hand crafted features.
- A novel regularization method for word sequences, dropword.
- Empirical evidence that highlights the importance of word order for WSD.

2 Background

2.1 Long short-term memory

Long short-term memory (LSTM) is a gated type of recurrent neural network (RNN). LSTMs were introduced by


Hochreiter and Schmidhuber (1997) to enable RNNs to better capture long term dependencies when used to model sequences. This is achieved by letting the model copy the state between timesteps without forcing the state through a non-linearity. The flow of information is instead regulated using multiplicative gates,

\[
i_n = \sigma(W^{x_i}x_n + W^{h_i}h_{n-1} + W^{c_i}c_{n-1} + b_i) \\
f_n = \sigma(W^{x_f}x_n + W^{h_f}h_{n-1} + W^{c_f}c_{n-1} + b_f) \\
o_n = \sigma(W^{x_o}x_n + W^{h_o}h_{n-1} + W^{c_o}c_{n-1} + b_o)
\]

the state of the network is saved in a memory cell,

\[
c_n = f_nc_{n-1} + i_n \tanh(W^{x_c}x_n + W^{h_c}h_{n-1} + b_c)
\]

and the output

\[
h_n = o_n \tanh(c_n)
\]

is computed using the state of the memory cell (c_n) and the multiplicative gate (o_n). \(b_{[i,f,o,c]}\) are bias units and \(\sigma\) is the logistic function. The behavior of all gates are learned from data together with the rest of the model, and captured in the model parameters: \(\Theta = \{W^{x_i}, W^{x_f}, W^{x_o}, W^{h_c}, W^{x_c}, b_c, W^{h_i}, W^{c_i}, b_i, W^{h_f}, W^{c_f}, b_f, W^{h_o}, W^{c_o}, b_o\}\).

2.2 Bidirectional LSTM

The bidirectional variant of LSTM, (BLSTM) (Graves and Schmidhuber, 2005) is a straightforward adaptation of the LSTM described in Section [2.1] where the state at each time step consist of the state of two LSTMs, one going left and one going right. For WSD this means that the state has information about both preceding words and succeeding words, which in many cases is absolutely necessary to correctly classify the sense.

2.3 Global vectors for word representation

Global Vectors for Word Representation (GloVe), introduced by Pennington et al. (2014) is a hybrid approach to word representation that combine a log-linear model, made popular by Mikolov et al. (2013), with counting based co-occurrence statistics to more efficiently capture global statistics. GloVe vectors are trained in an unsupervised fashion, typically on large amounts of data, and is able to capture fine grained semantic and syntactic information about words. These vectors can subsequently be used to initialize the input layer of a neural network or some other NLP model.

2.4 Regularization

When training powerful nonlinear classifiers like neural networks, good regularization is essential for generalization. One of the most basic regularization techniques is to add Gaussian noise to the inputs (Sijtsma and Dow, 1991). This can be viewed as a type of data augmentation, i.e. distorting the training data to create slightly different examples for each pass through the data, making it harder for the model to simply memorize the training set. Another, more recent noise based regularization technique is dropout, introduced by Srivastava et al. (2014). Dropout lowers the capacity of the model by randomly erasing information from a layer of neurons by setting a subset of them to zero. This forces the network to learn independent sub networks which improves the generalization in a similar way as to train many separate models and combining the results.

3 Dropword

Besides the regularization techniques described in Section [2.4] we present a novel regularizer referred to as dropword. The idea behind dropword is similar to that of dropout but instead of randomly removing a part of the activations, we randomly replace some words in the context with a <dropped> tag. The tag is treated just like any other word in the vocabulary and has a corresponding word embedding that is trained. This process is repeated over time, so that the words dropped change with time. The motivation for doing dropword is to decrease the dependency on individual words in the training context. This technique can be generalized to other kinds of sequential inputs, not only words.

4 The Model

Given a document and the position of the target word, i.e. the word to disambiguate, the model computes a probability distribution over the possible senses corresponding to that word. The architecture of the model, depicted in Figure 1, consist of a softmax layer, a hidden layer, and a BLSTM. See Section [2.2] for more details regarding the BLSTM. The BLSTM and the hidden layer share parameters over all word types and senses, while the softmax is parameterized by word type and selects the corresponding weight matrix and bias vector for each word type respectively. This structure enables the model to share statistical strength.
across different word types while remaining computationally efficient even for a large total number of senses.

### 4.1 Model definition

The input to the BLSTM at position \( n \) in document \( D \) is computed as

\[
x_n = W^x v(w_n), \quad n \in \{1, \ldots, |D|\}.
\]

where

\[
x_n = W^x v(w_n), \quad n \in \{1, \ldots, |D|\}.
\]

Here, \( v(w_n) \) is the one-hot representation of the word type corresponding to \( w_n \in D \). This will have the effect of picking the column from \( W^x \) corresponding to that word type. The resulting vector is referred to as word embedding. Further, \( W^x \) can be initialized using pre-trained word embeddings, to leverage large unannotated datasets.

The parameters of the model, \( \Omega \) = \{\( W^z \), \( \Theta_{BLSTM} \), \( W^{ha} \), \( b^{ha} \), \( \{W^a_{w_i}, b^a_{w_i}\} \forall w_i \in V \}, \} are fitted by minimizing the cross entropy error

\[
L(\mathcal{I}; \Omega) = - \sum_{i \in \mathcal{I}} \sum_{j \in S(w_i)} t_{ij} \log y_j(i)
\]

where \( y_j(i) \) is the output of the sense classifier consisting of one hidden layer with linear units and a softmax. The softmax selects the corresponding weight matrix and bias vector for the word at position \( n \).

### 4.2 Loss function

The parameters of the model, \( \Omega = \{W^z, \Theta_{BLSTM}, W^{ha}, b^{ha}, \{W^a_{w_i}, b^a_{w_i}\} \forall w_i \in V \}, \} are fitted by minimizing the cross entropy error

\[
L(\mathcal{I}; \Omega) = - \sum_{i \in \mathcal{I}} \sum_{j \in S(w_i)} t_{ij} \log y_j(i)
\]

over a set of sense labeled tokens with indices \( \mathcal{I} \subset \{1, \ldots, |\mathcal{C}|\} \) within a training corpus \( \mathcal{C} \), each labeled with a target sense distribution \( t_{ij}, \forall i \in \mathcal{I} \).

### 5 Experiments

#### 5.1 Data and Task

To evaluate our proposed model we performed the lexical sample task of SensEval 2 (SE2) and SensEval 3 (SE3). For both of these tasks training and test data is supplied, as well as a standardized scorer to ensure fair comparisons.

#### 5.2 Experimental settings

A big challenge with these relatively small datasets is the risk of overfitting the model. To mitigate this problem we apply a 50% dropout to the word embeddings, the LSTM hidden state \( h \), and the hidden layer \( a \). Further, GloVe vectors trained on Wikipedia and Gigaword are used to initialize the word embeddings, and we use 10% dropword, 74+74 neurons in the BLSTM, embedding size of 100 and additive gaussian noise with zero mean and a standard deviation of \( 0.2 \sigma_i \), where \( \sigma_i \) is the standard deviation in embedding dimension \( i \) for all words in the embeddings matrix, \( W^x \). \( \sigma_i \) is updated after each weight update. To speed up training, we use a maximum context window length of 140 words. Allowing the model to use more distant data than this did not improve the results. Finally, to showcase the generality of the model we use identical hyperparameter settings for both SE2 and SE3 during test.

---

**Figure 1:** A BLSTM centered around a word at position \( n \). Its output is fed to a neural network sense classifier consisting of one hidden layer with linear units and a softmax. The softmax selects the corresponding weight matrix and bias vector for the word at position \( n \).
5.3 Training details

The model is implemented in *TensorFlow* (Abadi et al., 2015), and optimized using *Stochastic Gradient Descent* with momentum (0.1) and batch size 100. The learning rate is initialized to 2.0 and then decreased exponentially with a decay factor of 0.96. All parameters in the model are initialized in $U(-0.1, 0.1)$, with the exception of the biases that are initialized to zero. The data is cleaned by removing various tags. We do not remove uncommon words, but to keep the size of the vocabulary manageable we replaced numbers with a `<number>` tag. Words not in GloVe is initialized from $N(0, 0.1)$. This resulted in a vocabulary size of $|V| = 50817$ for SE2 and $|V| = 37998$ for SE3. Words not present in the training set is considered as unknown words during test.

5.4 Results

The results on the corresponding datasets can be seen in Table 1. 100JHU(R) was developed by Yarowsky et al. (2001) and achieved the best score on the English lexical sample task of SE2 with a F1 score of 64.2. Their system utilized a rich feature space based on raw words, lemmas, POS tags, bag-of-words, bi-gram, and tri-gram collocations, etc. as inputs to an ensemble classifier. Correspondingly, htsa3 by Grozea (2004) was the winner of the SE3 lexical sample task with a F1 score of 72.9. This system also used a rich set of features. These were used as inputs to a regularized least square classifier. IMS+adapted CW is a more recent system by Taghipour and Ng (2015) that introduces word embeddings in WSD, albeit separately trained, as a feature together with, again, a rich set of features including POS tags, collocations and surrounding words. A support vector machine is subsequently trained on the extracted features for each target word.

Our proposed model outperform previous state-of-the-art on SE2 and are tied with IMS+adapted CW on SE3. Moreover, we see that dropword consistently improves the results on both SE2 and SE3. Randomizing the order of the input words yields a substantially worse result, which provides evidence for our hypothesis that the order of the words are significant. We also see that the system effectively makes use of the information in the pre-trained word embeddings and that they are essential to the performance of our system on these datasets.

| Method                        | F1 score |
|-------------------------------|----------|
|                               | SE2  | SE3  |
| BLSTM (our proposed model)    | 66.9 | 73.4 |
| 100JHU(R)                     | 64.2 | -    |
| htsa3                         | -    | 72.9 |
| IMS+adapted CW                | 66.2 | 73.4 |
| BLSTM without dropword        | 66.5 | 72.9 |
| BLSTM without GloVe           | 54.6 | 59.0 |
| BLSTM, randomized word order  | 58.8 | 64.7 |

Table 1: Results for Senseval 2 and 3 on the English lexical sample task.

6 Conclusions & future work

We presented a BLSTM based model for WSD that was able to effectively exploit word order and achieves *state-of-the-art* results on two WSD dataset. The model stands in stark contrast to previous work in that it is an end-to-end trainable system, all the way from the text to the sense prediction. Further, we showed that employing pre trained word vectors improved the generalization power of the model. However, it is likely that this effect would diminish if the model was trained on more sense tagged data, which is an experiment we hope to conduct as future work. Also, in part due to the limited amount of data, we developed the regularization technique dropword that we could show improved the results under these constrained circumstances. Another effect that stemmed from the limited data is that we needed to scale down and regularize the model heavily. Therefore we hold it as likely that it is possible to improve on the reported results simply by scaling up the model and training on more data.

Acknowledgments

The authors would like to acknowledge the project *Towards a knowledge-based culturomics* supported by a framework grant from the Swedish Research Council (2012–2016; dnr 2012-5738).

References

[Abadi et al.2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean,
Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Petar Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Bovi et al.2015] Claudio Delli Bovi, Luis Espinosa-Anke, and Roberto Navigli. 2015. Knowledge base unification via sense embeddings and disambiguation. In Proceedings of EMNLP, pages 726–736.

[Graves and Schmidhuber2005] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks, 18(5):602–610.

[Grozea2004] Cristian Grozea. 2004. Finding optimal parameter settings for high performance word sense disambiguation. In Proceedings of Senseval-3 Workshop.

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

[Johansson and Pina2015] Richard Johansson and Luis Nieto Pina. 2015. Combining relational and distributional knowledge for word sense disambiguation. Nordic Conference of Computational Linguistics NODALIDA 2015, page 69.

[Mikolov et al.2013] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In HLT-NAACL, pages 746–751.

[Navigli2009] Roberto Navigli. 2009. Word Sense Disambiguation: a survey. ACM Computing Surveys, 41(2):1–69.

[Neelakantan et al.2015] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. arXiv preprint arXiv:1504.06654.

[Nieto Piña and Johansson2015] Luis Nieto Piña and Richard Johansson. 2015. A simple and efficient method to generate word sense representations. In Proceedings of Recent Advances in Natural Language Processing, pages 465–472, Hissar, Bulgaria.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014), 12:1532–1543.

[Sietsma and Dow1991] Jocelyn Sietsma and Robert JF Dow. 1991. Creating artificial neural networks that generalize. Neural networks, 4(1):67–79.

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

[Taghipour and Ng2015] Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In The 2015 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 314–323.

[Vickrey et al.2005] David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Wordsense disambiguation for machine translation. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 771–778. Association for Computational Linguistics.

[Yarowsky et al.2001] David Yarowsky, Silviu Cucerzan, Radu Florian, Charles Schafer, and Richard Wicentowski. 2001. The johns hopkins senseval2 system descriptions. In The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems, pages 163–166. Association for Computational Linguistics.