Word Sense Disambiguation using a Bidirectional LSTM

Mikael Kågebäck[*], Hans Salomonsson[*]
Computer Science & Engineering, Chalmers University of Technology
SE-412 96, Göteborg, Sweden
{kageback,hans.salomonsson}@chalmers.se

Abstract

In this paper we present a clean, yet effective, model for *word sense disambiguation*. Our approach leverage a *bidirectional long short-term memory* network which is shared between all words. This enables the model to share statistical strength and to scale well with vocabulary size. The model is trained *end-to-end*, directly from the raw text to sense labels, and makes effective use of word order. We evaluate our approach on two standard datasets, using identical hyperparameter settings, which are in turn tuned on a third set of held out data. We employ no external resources (e.g. knowledge graphs, part-of-speech tagging, etc), language specific features, or hand crafted rules, but still achieve statistically equivalent results to the best *state-of-the-art* systems, that employ no such limitations.

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 of *rock* is no longer ambiguous. The task of assigning a word token in a text, e.g. *rock*, to a well defined word sense in a lexicon is called *word sense disambiguation* (WSD). From the *rock* example above it is easy to see that the context surrounding the word is what disambiguates the sense. However, it may not be so obvious that this is a difficult task. To see this, consider instead the phrase "Solid rock" where changing the order of words completely changes the meaning, or "Hard rock crushes heavy metal" where individual words seem to indicate stone but together they actually define the word token as music. With this in mind, our thesis is that to do WSD well we need to go beyond *bag of words* and into the territory of sequence modeling.

Improved WSD would be beneficial to many natural language processing (NLP) problems, e.g. machine translation (Vickrey et al., 2005), information Retrieval, information Extraction (Navigli, 2009), and sense aware word representations (Neelakantan et al., 2015; Kågebäck et al., 2015; Nieto Piña and Johansson, 2015; Bovi et al., 2015). However, though much progress has been made in the area, many current WSD systems suffer from one or two of the following deficits. (1) Disregarding the order of words in the context which can lead to problems as described above. (2) Relying on complicated and potentially language specific hand crafted features and resources, which is a big problem particularly for resource poor languages. We aim to mitigate these problems by (1) modeling the sequence of words surrounding the target word, and (2) refrain from using any hand crafted features or external resources and instead represent the words using real valued vector representation, i.e. word embeddings. Using word embeddings has previously been shown to improve WSD (Taghipour and Ng, 2015; Johansson and Nieto Piña, 2015). However, these works did not consider the order of words or their operational effect on each other.

[*]Authors contributed equally.

1.1 The main contributions of this work include:

- A purely learned approach to WSD that achieves results on par with state-of-the-art resource heavy systems, employing e.g. knowledge graphs, parsers, part-of-speech tagging, etc.
- Parameter sharing between different word types to make more efficient use of labeled data and make full vocabulary scaling plausible without the number of parameters exploding.
- Empirical evidence that highlights the importance of word order for WSD.
- A WSD system that, by using no explicit window, is allowed to combine local and global information when deducing the sense.

2 Background

In this section we introduce the most important underlying techniques for our proposed model.

2.1 Bidirectional LSTM

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 which preserves the gradient better than e.g. the logistic function. The bidirectional variant of LSTM, (BLSTM) (Graves and Schmidhuber, 2005) is an adaptation of the LSTM 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 are absolutely necessary to correctly classify the sense.

2.2 Word embeddings by GloVe

Word embeddings is a way to represent words as real valued vectors in a semantically meaningful space. *Global Vectors for Word Representation* (GloVe), introduced by Pennington et al. (2014) is a hybrid approach to embedding words 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. Word embeddings 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.

3 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 2.1 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 and realistic vocabulary sizes.

3.1 Model definition

The input to the BLSTM at position n in document $\mathcal{D}$ is computed as

$$\mathbf{x}_n = W^x \mathbf{v}(w_n), n \in \{1, \dots, |\mathcal{D}|\}.$$

Here, $\mathbf{v}(w_n)$ is the *one-hot* representation of the word type corresponding to $w_n \in \mathcal{D}$. A one-hot representation is a vector with dimension V consisting of $|V| - 1$ zeros and a single one which index

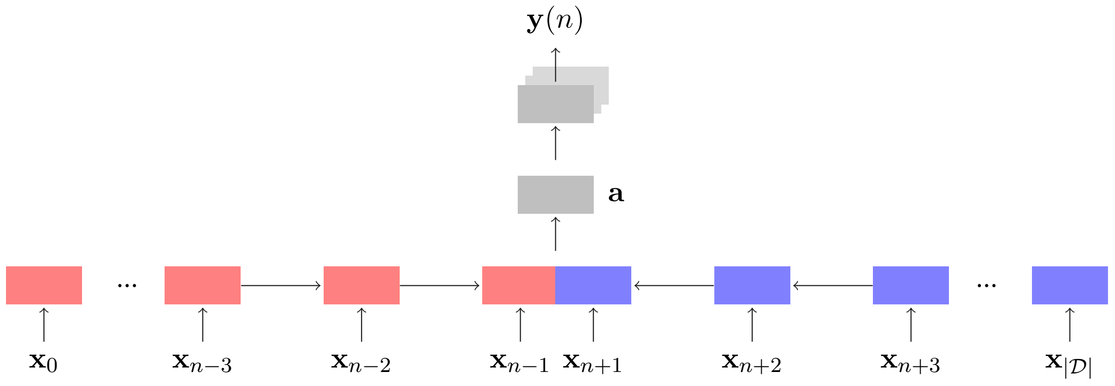

Figure 1: A BLSTM centered around a word at position n. Its output is fed to a neural network sense classifier consisting of one hidden layer with linear units and a softmax. The softmax selects the corresponding weight matrix and bias vector for the word at position n.

.

indicate the word type. This will have the effect of picking the column from W^x corresponding to that word type. The resulting vector is referred to as a word embedding. Further, W^x can be initialized using pre-trained word embeddings, to leverage large unannotated datasets. In this work GloVe vectors are used for this purpose, see Section 4.1 for details.

The model output,

$$\mathbf{y}(n) = \text{softmax}(W^{ay}_{w_n}\mathbf{a} + \mathbf{b}^{ay}_{w_n}),$$

is the predicted distribution over senses for the word at position n, where $W^{ay}_{w_n}$ and $\mathbf{b}^{ay}_{w_n}$ are the weights and biases for the softmax layer corresponding to the word type at position n. Hence, each word type will have its own softmax parameters, with dimensions depending on the number of senses of that particular word. Further, the hidden layer $\mathbf{a}$ is computed as

$$\mathbf{a} = W^{ha}[\mathbf{h}^L_{n-1}; \mathbf{h}^R_{n+1}] + \mathbf{b}^{ha}$$

where $[\mathbf{h}^L_{n-1}; \mathbf{h}^R_{n+1}]$ is the concatenated outputs of the right and left traversing LSTMs of the BLSTM at word n. W^{ha} and $\mathbf{b}^{ha}$ are the weights and biases for the hidden layer.

Loss function The parameters of the model, $\Omega = \{W^x, \Theta_{BLSTM}, W^{ha}, \mathbf{b}^{ha}, \{W^{ay}_w, \mathbf{b}^{ay}_w\}_{\forall w \in V}, \}$, are fitted by minimizing the cross entropy error

$$L(\Omega) = -\sum_{i \in \mathcal{I}} \sum_{j \in S(w_i)} t_{i,j} \log y_j(i)$$

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 $\mathbf{t}_i, \forall i \in \mathcal{I}$.

3.2 Dropword

Dropword is a regularization technique very similar to *word dropout* introduced by Iyyer et al. (2015). Both methods are word level generalizations of dropout (Srivastava et al., 2014) but in word dropout the word is set to zero while in dropword it is replaced with a <*dropped*> tag. The tag is subsequently 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 over 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 Experiments

To evaluate our proposed model we perform the *lexical sample task* of SensEval 2 (SE2) (Kilgarriff, 2001) and SensEval 3 (SE3) (Mihalcea et al., 2004), part of the SensEval (Kilgarriff and Palmer, 2000) workshops organized by *Special Interest Group on the Lexicon* at ACL. For both instances of the task training and test data are supplied, and the task consist of disambiguating one indicated word in a context. The words to disambiguate are sampled from the vocabulary to give a range of low, medium and high frequency words, and a gold standard sense label is supplied for training and evaluation.

4.1 Experimental settings

The hyperparameter settings used during the experiments, presented in Table 1, were tuned on a separate validation set with data picked from the SE2 training set. The source code, implemented using *TensorFlow* (Abadi et al., 2015), has been released as open source[1].

Hyperparameter	**Range searched**	**Value used**
Embedding size	{100, 200}	100
BLSTM hidden layer size	[50, 100]	$2 * 74$
Dropout on word embeddings $\mathbf{x}_n$	[0, 50%]	50%
Dropout on the LSTM output $[\mathbf{h}^L_{n-1}; \mathbf{h}^R_{n+1}]$	[0, 70%]	50%
Dropout on the hidden layer $\mathbf{a}$	[0, 70%]	50%
Dropword	[0, 20%]	10%
Gaussian noise added to input	[0, 0.4]	$\sim \mathcal{N}(0, 0.2\sigma_i)$
Optimization algorithm	-	Stochastic gradient descent
Momentum	-	0.1
Initial learning rate	-	2.0
Learning rate decay	-	0.96
Embedding initialization	-	GloVe
Remaining parameters initialized	-	$\in \mathcal{U}(-0.1, 0.1)$

Table 1: Hyperparameter settings used for both experiments and the ranges that were searched during tuning. ”-” indicates that no tuning were performed on that parameter.

Embeddings The embeddings are initialized using a set of freely available[2] GloVe vectors trained on Wikipedia and Gigaword. Words not included in this set are initialized from $\mathcal{N}(0, 0.1)$. To keep the input noise proportional to the embeddings it is scaled by σ_i which is the standard deviation in embedding dimension i for all words in the embeddings matrix, W^x. σ_i is updated after each weight update.

Data preprocessing The only preprocessing of the data that is conducted is replacing numbers with a $< number >$ tag. This result in a vocabulary size of $|V| = 50817$ for SE2 and $|V| = 37998$ for SE3. Words not present in the training set are considered unknown during test. Further, we limit the size of the context to max 140 words centered around the target word to facilitate faster training.

4.2 Results

The results of our experiments and the state-of-the-art are shown in Table 2. 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. htsa3 by Grozea (2004) was the winner of the SE3 lexical sample task with a F1 score of 72.9. This system was based mainly on raw words, lemmas, and POS tags. These were used as inputs to a regularized least square

[1]Source for all experiments is available at: `https://bitbucket.org/salomons/wsd`

[2]The employed GloVe vectors are available for download at: `http://nlp.stanford.edu/projects/glove/`

classifier. IMS+adapted CW is a more recent system, by Taghipour and Ng (2015), that uses separately trained word embeddings as input. However, it also relies on a rich set of other features including POS tags, collocations and surrounding words to achieve their reported result.

Our proposed model achieves the top score 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.

	F1 score	
Method	**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 2: Results for Senseval 2 and 3 on the English lexical sample task.

5 Conclusions & future work

We presented a BLSTM based model for WSD that was able to effectively exploit word order and achieve results on *state-of-the-art* level, using no external resources or handcrafted features. As a consequence, the model is largely language independent and applicable to resource poor languages. Further, the system was designed to generalize to full vocabulary WSD by sharing most of the parameters between words.

For future work we would like to provide more empirical evidence for language independence by evaluating on several different languages, and do experiments on large vocabulary *all words WSD*, where every word in a sentence is disambiguated. Further, we plan to experiment with unsupervised pre-training of the BLSTM, encouraged by the substantial improvement achieved by incorporating word embeddings.

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). Further, the work was partly funded by the *BigData@Chalmers* initiative. Finally. the authors would like to thank the reviewers for their insightful and constructive feedback.

References

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, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

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.

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

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

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*.

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

Mikael Kågebäck, Fredrik Johansson, Richard Johansson, and Devdatt Dubhashi. 2015. Neural context embeddings for automatic discovery of word senses. In *Proceedings of NAACL-HLT*, pages 25–32.

Adam Kilgarriff and Martha Palmer. 2000. Introduction to the special issue on senseval. *Computers and the Humanities*, 34(1-2):1–13.

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

Rada Mihalcea, Timothy Anatolievich Chklovski, and Adam Kilgarriff. 2004. The senseval-3 english lexical sample task. Association for Computational Linguistics.

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

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

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*.

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.

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

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.

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.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word-sense 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.

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*, SENSEVAL '01, pages 163–166, Stroudsburg, PA, USA. Association for Computational Linguistics.