Semi-Supervised Word Sense Disambiguation Using Word Embeddings in General and Specific Domains

Kaveh Taghipour
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
kaveh@comp.nus.edu.sg

Hwee Tou Ng
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
nght@comp.nus.edu.sg

Abstract

One of the weaknesses of current supervised word sense disambiguation (WSD) systems is that they only treat a word as a discrete entity. However, a continuous-space representation of words (word embeddings) can provide valuable information and thus improve generalization accuracy. Since word embeddings are typically obtained from unlabeled data using unsupervised methods, this method can be seen as a semi-supervised word sense disambiguation approach. This paper investigates two ways of incorporating word embeddings in a word sense disambiguation setting and evaluates these two methods on some SensEval/SemEval lexical sample and all-words tasks and also a domain-specific lexical sample task. The obtained results show that such representations consistently improve the accuracy of the selected supervised WSD system. Moreover, our experiments on a domain-specific dataset show that our supervised baseline system beats the best knowledge-based systems by a large margin.

1 Introduction

Because of the ambiguity of natural language, many words can have different meanings in different contexts. For example, the word "bank" has two different meanings in "the bank of a river" and "a bank loan". While it seems simple for humans to identify the meaning of a word according to the context, word sense disambiguation (WSD) (Ng and Lee, 1996; Lee and Ng, 2002) is a difficult task for computers and thus requires sophisticated means to achieve its goal. Part of this ambiguity may be resolved by considering part-of-speech (POS) tags but the word senses are still highly ambiguous even for the same part-of-speech. Machine translation is probably the most important application of word sense disambiguation. In machine translation, different senses of a word cause a great amount of ambiguity for automated translation and it negatively affects the results. Hence, an accurate WSD system can benefit machine translation significantly and improve the results (Chan et al., 2007; Carpuat and Wu, 2007; Vickrey et al., 2005). Moreover, Zhong and Ng (2012) have shown that word sense disambiguation improves information retrieval by proposing a method to use word senses in a language modeling approach to information retrieval.

The rest of this paper is organized as follows. Section 2 gives a literature review of related work, including a review of semi-supervised word sense disambiguation and distributed word representation called word embeddings. The method and framework used in this paper are explained in Section 3. Finally, we evaluate the system in Section 4 and conclude the paper in Section 5.

2 Related Work

The method that we use in this paper is a semi-supervised learning method which incorporates knowledge from unlabeled datasets by using word embeddings. This section is a literature review of previous work on semi-supervised word sense disambiguation and various methods of obtaining word embeddings.

Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL, pages 314–323,
Denver, Colorado, May 31 – June 5, 2015.

2.1 Semi-Supervised Word Sense Disambiguation

Among various types of semi-supervised learning approaches, *co-training* and *self-training* are probably the most common. These methods randomly select a subset of a large unlabeled dataset and classify these samples using one (self-training) or two (co-training) classifiers, trained on a smaller set of labeled samples. After assigning labels to the new samples, these methods select the samples that were classified with a high confidence (according to a selection criterion) and add them to the set of labeled data. These methods have been used in the context of word sense disambiguation. Mihalcea (2004) used both co-training and self-training to make use of unlabeled datasets for word sense disambiguation. Mihalcea also introduced a technique for combining co-training and majority voting, called *smoothed co-training*, and reported improved results. Another related study was done by (Pham et al., 2005). In (Pham et al., 2005), some semi-supervised learning techniques were used for word sense disambiguation. Pham et al. employed co-training and spectral graph transduction methods in their experiments and obtained significant improvements over a supervised method.

Another semi-supervised learning method used for word sense disambiguation is Alternating Structure Optimization (ASO), first introduced by (Ando and Zhang, 2005) and later applied to word sense disambiguation tasks by (Ando, 2006). This algorithm learns a predictive structure shared between different problems (disambiguation of a target word). Semi-supervised application of the ASO algorithm was shown to be useful for word sense disambiguation and improvements can be achieved over a supervised predictor (Ando, 2006).

This paper uses a different method proposed by (Turian et al., 2010) that can be applied to a wide variety of supervised tasks in natural language processing. This method uses distributed word representations (word embeddings) as additional feature functions in supervised tasks and is shown to improve the accuracy of named-entity recognition (NER) and chunking. In this paper, we also follow the same approach for word sense disambiguation. The key idea is that a system without a continuous-space representation of words ignores the similarity of words completely and relies only on their discrete form. However, when a distributed representation for words is added to the system, the classifier can make use of the notion of *similarity* of words and learn the relationships between class labels and words.

In addition to using *raw* word embeddings, we also propose a method to *adapt* embeddings for each classification task. Since word embeddings do not include much task-specific discriminative information, we use a neural network to modify word vectors to tune them for our WSD tasks. We show that this process results in improved accuracy compared to raw word embeddings.

Recently, obtaining word embeddings in an unsupervised manner from large text corpora has attracted the attention of many researchers (Collobert and Weston, 2008; Mnih and Hinton, 2009; Mikolov et al., 2013a; Mikolov et al., 2013b). Subsequently, there have been some published word embeddings and some software for training word embeddings.

For word sense disambiguation, there are very few open source programs. Since we are interested in a fully supervised WSD tool, IMS (It Makes Sense) (Zhong and Ng, 2010) is selected in our work. This system allows addition of extra features in a simple way and hence is a good choice for testing the effect of word embeddings as additional features. Moreover, the scores reported for IMS are competitive with or better than state-of-the-art systems (Zhong and Ng, 2010).

2.2 Word Embeddings

There are several types of word representations. A *one-hot* representation is a vector where all components except one are set to zero and the component at the index associated with a word is set to one. This type of representation is the sparsest word representation and does not carry any information about word similarity. Another popular approach is to use the methods mainly applied in information retrieval. Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA) are such examples, and word representations produced by these methods can also be used in other applications. However, a dense distributed representation for words (word embeddings) can learn more complex relationships

between words and hence, it can be useful in a wide range of applications. We only focus on word embeddings in this paper and apply them to word sense disambiguation.

Word embeddings are distributed representations of words and contain some semantic and syntactic information (Mikolov et al., 2013c). Such representations are usually produced by neural networks. Examples of such neural networks are (log-)linear networks (Mikolov et al., 2013a), deeper feed-forward neural networks (Bengio et al., 2003; Collobert and Weston, 2008), or recurrent neural networks (Mikolov et al., 2010). Moreover, it has been shown that deep structures may not be needed for word embeddings estimation (Lebret et al., 2013) and shallow structures can obtain relatively high quality representations for words (Mikolov et al., 2013b).

In this paper, we have used the word embeddings created and published by (Collobert and Weston, 2008). Throughout this paper, we refer to these word embeddings as 'CW'. This method is proposed in (Collobert and Weston, 2008) and explained further in (Collobert et al., 2011). The authors use a feed-forward neural network to produce word representations. In order to train the neural network, a large text corpus is needed. Collobert and Weston (2008) use Wikipedia (Nov. 2007 version containing 631 million words) and Reuters RCV1 (containing 221 million words) (Lewis et al., 2004) as their text corpora and Stochastic Gradient Descent (SGD) as the training algorithm. The training algorithm selects a window of text randomly and then replaces the middle word with a random word from the dictionary. Then the original window of text and the corrupted one is given to the neural network. The neural network computes $f(x)$ and $f(x^{(w)})$, where x is the original window of text, $x^{(w)}$ is the same window of text with the middle word replaced by word w, and $f(.)$ is the function that the neural network represents. After computing $f(x)$ and $f(x^{(w)})$, the training algorithm uses a *pairwise ranking* cost function to train the network. The training algorithm minimizes the cost function by updating the parameters (including word embeddings) and as a consequence of using the pairwise ranking cost function, this neural network tends to assign higher scores to *valid* windows of texts and lower scores to *incorrect* ones. After training the neural network, the word vectors in the lookup table layer form the word embeddings matrix. Collobert et al. (2011) have made this matrix available for public use and it can be accessed online[1].

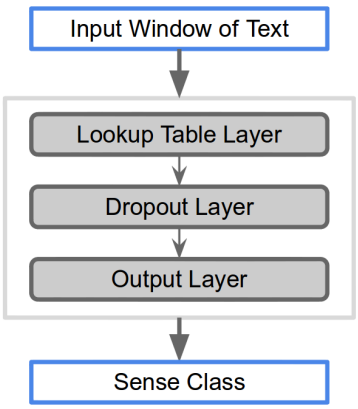

Figure 1: The neural network architecture for adaptation of word embeddings.

3 Method

In this section, we first explain our novel task-specific method of *adapting* word embeddings and then describe our framework in which raw or adapted word embeddings are included in our word sense disambiguation system. To the best of our knowledge, the use of word embeddings for semi-supervised word sense disambiguation is novel.

3.1 Adaptation of Word Embeddings

Word embeddings capture some semantic and syntactic information and usually similar words have similar word vectors in terms of distance measures. However, in a classification task, it is better for word embeddings to also include some task specific discriminative information. In order to add such information to word embeddings, we modify word vectors using a neural network (Figure 1) to obtain *adapted* word embeddings. This section explains this process in detail.

The neural network that we used to adapt word embeddings is similar to the *window approach* network introduced by (Collobert and Weston, 2008). This neural network includes the following layers:

[1] http://ml.nec-labs.com/senna

- Lookup table layer: This layer includes three lookup tables. The first lookup table assigns a vector to each input word, as described earlier. The second lookup table maps each word to a vector with respect to the capitalization feature. Finally, the third lookup table maps each word to its corresponding vector based on the word's POS tag.

- Dropout layer: In order to avoid overfitting, we added a dropout layer (Hinton et al., 2012) to the network to make use of its regularization effect. During training, the dropout layer copies the input to the output but randomly sets some of the entries to zero with a probability p, which is usually set to 0.5. During testing, this layer produces the output by multiplying the input vector by $1 - p$ (see (Hinton et al., 2012) for more information).

- Output layer: This layer linearly maps the input vector X to a C-dimensional vector Y (equation 1) and then applies a SoftMax operation (Bridle, 1990) over all elements of Y (equation 2):

$$Y = WX + b \quad (1)$$

$$p(t|I) = \frac{\exp(Y_t)}{\sum_{j=1}^{C} \exp(Y_j)} \qquad 1 \leq t \leq C \quad (2)$$

  where I is the input window of text and t is a class label (sense tag in WSD). The output of the output layer can be interpreted as a conditional probability $p(t|I)$ over tags given the input text.

This architecture is similar to the network used by (Collobert and Weston, 2008) but it does not include a hidden layer. Since the number of training samples for each word type in WSD is relatively small, we did not use a hidden layer to decrease the model size and consequently overfitting, as much as possible. Moreover, we added the dropout layer and observed increased generalization accuracy subsequently.

In order to train the neural network, we used Stochastic Gradient Descent (SGD) and error back-propagation to minimize negative log-likelihood cost function for each training example (I, t) (equation 3).

$$-\log p(t|I) = \log(\sum_{j=1}^{C} exp(Y_j)) - Y_t \quad (3)$$

During the training process, the inputs are the windows of text surrounding the target word with their assigned POS tags. We used fixed learning rate (0.01) during training, with no momentum.

Since the objective is to adapt word embeddings using the neural network, we initialized the lookup table layer parameters using pre-trained word embeddings and trained a model for each target word type. After the training process completes, the modified word vectors form our adapted word embeddings, which will be used in exactly the same way as the original embeddings. Section 3.2 explains the way we use word embeddings to improve a supervised word sense disambiguation system.

3.2 Framework

The supervised system that we used for word sense disambiguation is an open source tool named IMS (Zhong and Ng, 2010). This software extracts three types of features and then uses Support Vector Machines (SVM) as the classifier. The three types of features implemented in IMS are explained below.

- POS tags of surrounding words: IMS uses the POS tags of all words in a window size of 7, surrounding the target ambiguous word. POS tag features are limited to the current sentence and neighboring sentences are not considered.

- Surrounding words: Additionally, the surrounding words of a target word (after removing stop words) are also used as features in IMS. However, unlike POS tags, the words occurring in the immediately adjacent sentences are also included.

- Local collocations: Finally, 11 local collocations around the target word are considered as features. These collocations also cover a window size of 7, where the target word is in the middle.

All mentioned features are binary features and will be used by the classifier in the next phase. After extracting these features, the classifier (SVM) is

used to train a model for each target word. In the test phase, the model is used to classify test samples and assign a sense tag to each sample.

This supervised framework with separate feature extraction and classification phases makes it easy to add any number of features and in our case, word embeddings. In order to make use of word embeddings trained by a neural network, we follow the approach of (Turian et al., 2010) and include word embeddings for all words in the surrounding window of text, given a target ambiguous word. We use the words from immediately adjacent sentences if the window falls beyond the current sentence boundaries. Since each word type has its own classification model, we do not add the word embeddings for the target word because the same vector will be used in all training and test samples and will be useless.

After extraction of the three mentioned types of features, $d.(w-1)$ features will be added to each sample, where d is the word embeddings dimension and w is the number of words in the window of text surrounding the target word (window size). w is one of the hyper-parameters of our system that can be tuned for each word type separately. However, since the training sets for some of the benchmark tasks are small, tuning the window size will not be consistent over different tuning sets. Thus, we decided to select the same window size for all words in a task and tune this parameter on the whole tuning set instead. After augmenting the features, a model is trained for the target word and then the classifier can be used to assign the correct sense to each test sample.

However, since the original three types of features are binary, newly added real-valued word embeddings do not fit well into the model and they tend to decrease performance. This problem is addressed in (Turian et al., 2010) and a simple solution is to scale word embeddings. The following conversion is suggested by (Turian et al., 2010):

$$E \leftarrow \sigma \cdot E / stddev(E) \quad (4)$$

where σ is a scalar hyper-parameter denoting the desired standard deviation, E is the word embeddings matrix and $stddev(.)$ is the standard deviation function, which returns a scalar for matrix E. However, different dimensions of word embedding vectors may have different standard deviations and Equation 4 may not work well. In this case, per-dimension scaling will make more sense. In order to scale the word embeddings matrix, we use Equation 5 in our experiments:

$$E_i \leftarrow \sigma \cdot E_i / stddev(E_i), \quad i : 1, 2, ..., d \quad (5)$$

where E_i denotes the i^{th} dimension of word embeddings. Like (Turian et al., 2010), we also found that $\sigma = 0.1$ is a good choice for the target standard deviation and works well.

4 Results and Discussion

We evaluate our word sense disambiguation system experimentally by using standard benchmarks. The two major tasks in word sense disambiguation are *lexical sample task* and *all-words task*. For each task, we explain our experimental setup first and then present the results of our experiments for the two mentioned tasks. Although most benchmarks are general domain test sets, a few domain-specific test sets also exist (Koeling et al., 2005; Agirre et al., 2010).

4.1 Lexical Sample Tasks

We have evaluated our system on SensEval-2 (SE2) and SensEval-3 (SE3) lexical sample tasks and also the domain-specific test set (we call it DS05) published by (Koeling et al., 2005). This subsection describes our experiments and presents the results of these tasks.

4.1.1 Experimental Setup

Most lexical sample tasks provide separate training and test sets. Some statistics about these tasks are given in Table 1.

	SE2	**SE3**	**DS05**
#Word types	73	57	41
#Training samples	8,611	8,022	-
#Test samples	4,328	3,944	10,272

Table 1: Statistics of lexical sample tasks

The DS05 dataset does not provide any training instances. In order to train models for DS05 (and later for the SE3 all-words task), we generated training samples for the top 60% most frequently occurring polysemous content words in Brown Corpus,

using the approach described in (Ng et al., 2003; Chan and Ng, 2005). This dataset is automatically created by processing parallel corpora without any manual sense annotation effort. We used the following six English-Chinese parallel corpora: Hong Kong Hansards, Hong Kong News, Hong Kong Laws, Sinorama, Xinhua News, and the English translations of Chinese Treebank. Similar to (Zhong and Ng, 2010), we obtained word alignments using GIZA++ (Och and Ney, 2000). Then, for each English word, the aligned Chinese word is used to find the corresponding sense tag for the English word. Finally, we made use of examples from the DSO corpus (Ng and Lee, 1996) and SEMCOR (Miller et al., 1994) as part of our training data. Table 2 shows some statistics of our training data.

POS	#word types
Adj.	5,129
Adv.	28
Noun	11,445
Verb	4,705
Total	21,307

Table 2: Number of word types in each part-of-speech (POS) in our training set

Since the dataset used by (Zhong and Ng, 2010) does not cover the specific domains of DS05 (Sports and Finance), we added a few samples from these domains to improve our baseline system. For each target word, we *randomly* selected 5 instances (a sentence including the target word) for Sports domain and 5 instances for Finance domain from the Reuters (Rose et al., 2002) dataset's Sports and Finance sections and manually sense annotated them. Annotating 5 instances per word and domain takes about 5 minutes. To make sure that these instances are not the same samples in the test set, we filtered out all documents containing at least one of the test instances and selected our training samples from the rest of the collection. After removing samples with *unclear* tags, we added the remaining instances (187 instances for Sports domain and 179 instances for Finance domain) to our original training data (Zhong and Ng, 2010). We highlight this setting in our experiments by 'CC' (concatenation).

We used the published CW word embeddings and set the word embeddings dimension to 50 in all our experiments. Finally, in order to tune the window size hyper-parameter, we randomly split our training sets into two parts. We used 80% for training models and the remaining 20% for evaluation. After tuning the window size, we used the original complete training set for training our models.

4.1.2 Results

In order to select a value for the window size parameter, we performed two types of tuning. The first method, which (theoretically) can achieve higher accuracies, is per-word tuning. Since each word type has its own model, we can select different window sizes for different words. The second method, on the other hand, selects the same value for the window size for all word types in a task, and we call it per-task tuning.

Although, per-word tuning achieved very high accuracies on the held-out development set, we observed that it performed poorly on the test set. Moreover, the results of per-word tuning are not stable and different development sets lead to different window sizes and also fluctuating accuracies. This is because the available training sets are small and using 20% of these samples as the development set means that the development set only contains a small number of samples. Thus the selected development sets are not proper representatives of the test sets and the tuning process results in overfitting the parameters (window sizes) to the development sets, with low generalization accuracy. However, per-task tuning is relatively stable and performs better on the test sets. Thus we have selected this method of tuning in all our experiments. Mihalcea (2004) also reports that per-word tuning of parameters is not helpful and does not result in improved performance.

We also evaluated our system separately on the word types in each part-of-speech (POS) for SE2 and SE3 lexical sample tasks. The results are included in Table 3 and Table 4. According to these tables, word embeddings do not affect all POS types uniformly. For example, on SE2, the improvement achieved on verbs is much larger than the other two POS types and on SE3, adjectives benefited from word embeddings more than nouns and verbs. However, this table also shows that improvements from word embeddings are consistent over POS types and

both lexical sample tasks.

	SE2		
POS	**#word types**	**baseline**	**CW (17)**
Adj.	15	67.45%	67.72%
Noun	29	69.39%	69.38%
Verb	29	60.45%	61.89%

Table 3: The scores for each part-of-speech (POS) on SE2 lexical sample tasks. The window size is shown inside brackets.

	SE3		
POS	**#word types**	**baseline**	**CW (9)**
Adj.	5	45.93%	47.81%
Noun	32	73.44%	73.83%
Verb	20	74.12%	74.17%

Table 4: The scores for each part-of-speech (POS) on SE3 lexical sample tasks. The window size is shown inside brackets.

Finally, we evaluated the effect of word embeddings and the adaptation process. Table 5 summarizes our findings on SE2 and SE3 lexical sample tasks. According to this table, both types of word embeddings lead to improvements on lexical sample tasks. We also performed a one-tailed paired t-test to see whether the improvements are statistically significant over the baseline (IMS). The improvements obtained using CW word embeddings over the baseline are significant ($p < 0.05$) in both lexical sample tasks. Furthermore, the results show that adapted word embeddings achieve higher scores than raw word embeddings. We have included the scores obtained by the first and the second best participating systems in these lexical sample tasks and also the Most Frequent Sense (MFS) score. The results also show that the Dropout layer increases performance significantly. The reason behind this observation is that without Dropout, word embeddings are overfitted to the training data and when they are used as extra features in IMS, the classifier does not generalize well to the test set. Since adaptation without Dropout leads to worse performance, we include Dropout in all other experiments and only report results obtained using Dropout.

Similarly, Table 6 presents the results obtained from our experiments on the DS05 dataset. In this table, as explained earlier, 'CC' denotes the additional manually tagged instances. For comparison purposes, we included the results reported by two state-of-the-art knowledge-based systems, namely PPR_{w2w} (Agirre et al., 2014) and Degree (Ponzetto and Navigli, 2010).

	SE2	SE3
IMS (baseline)	65.3%	72.7%
IMS + CW	66.1%* (17)	73.0%* (9)
IMS + adapted CW	**66.2%* (5)**	**73.4%* (7)**
– Dropout	65.4% (7)	72.7% (7)
Rank 1 system	64.2%	72.9%
Rank 2 system	63.8%	72.6%
MFS	47.6%	55.2%

Table 5: Lexical sample task results. The values inside brackets are the selected window sizes and statistically significant ($p < 0.05$) improvements are marked with '*'.

Table 6 shows that IMS performs worse than PPR_{w2w} on Sports and Finance domains but IMS + CC outperforms PPR_{w2w}. One of the reasons behind this observation is *unseen* sense tags. For example, in the sentence "the winning goal came with less than a minute left to play[2]", the sense tag for word 'goal' is 'goal%1:04:00::'. However, the training data for IMS does not contain any sample with this sense tag and so it is impossible for IMS to assign this tag to any test instances. On the other hand, the manually annotated instances (CC) include samples with this tag and therefore IMS + CC is able to associate a target word with this sense tag.

According to Table 6, adding word embeddings results in improved performance over the baseline (IMS + CC). Moreover, adapting word embeddings is found to increase accuracy in most cases.

4.2 All-Words Task

We also evaluated the performance of our system on the SensEval-3 (SE3) all-words task. Next, we explain our setup and then present the results of our evaluation.

[2]This example is taken from WordNet v3.1.

	BNC	Sports	Finance	Total
IMS	48.7%	41.4%	53.4%	47.8%
IMS + CC (baseline)	51.7%	55.7%	62.1%	56.4%
IMS + CC + CW (3)	51.9%	56.1%*	**62.3%**	56.7%*
IMS + CC + adapted CW (3)	**52.3%***	**57.1%***	62.0%	**57.1%***
PPR_{w2w}	37.7%	51.5%	59.3%	49.3%
Degree	-	42.0%	47.8%	-

Table 6: DS05 task results. The values inside brackets are the selected window sizes and statistically significant ($p < 0.05$) improvements over 'IMS + CC' are marked with '*'.

4.2.1 Experimental Setup

All-words tasks do not provide any training samples and only include a test set (see Table 7). In order to train our system for SE3 all-words task, we used the automatically labeled training samples used earlier for training models for DS05 (see section 4.1.1). Table 2 shows some statistics about our training set.

	SE3
#Word types	963
#Test samples	2,041

Table 7: Statistics of SE3 all-words task

Similar to the lexical sample tasks, we tune our system on 20% of the original training set. After obtaining window size parameter via tuning, we train on the whole training set and test on the given standard test set.

4.2.2 Results

The results of the evaluation on SE3 all-words task are given in Table 8. This table shows that CW word embeddings improve the accuracy. Similar to the results obtained for the lexical sample tasks, we observe some improvement by adapting word embeddings for SE3 all-words task as well. For comparison purposes, we have included the official scores of rank 1 and rank 2 participating systems in SE3 all-words task and the WordNet first sense (WNs1) score.

	SE3
IMS (baseline)	67.6%
IMS + CW	68.0%* (9)
IMS + adapted CW	**68.2%* (9)**
Rank 1 system	65.2%
Rank 2 system	64.6%
WNs1	62.4%

Table 8: SE3 all-words task results. The values inside brackets are the selected window sizes and statistically significant ($p < 0.05$) improvements over the IMS baseline are marked with '*'.

5 Conclusion

Supervised word sense disambiguation systems usually treat words as discrete entities and consequently ignore the concept of *similarity* between words. However, by adding word embeddings, some of the samples that cannot be discriminated based on the original features (surrounding words, collocations, POS tags) have more chances to be classified correctly. Moreover, word embeddings are likely to contain valuable linguistic information too. Hence, adding continuous-space representations of words can provide valuable information to the classifier and the classifier can learn better discriminative criteria based on such information.

In this paper, we exploited a type of word embeddings obtained by feed-forward neural networks. We also proposed a novel method (i.e., adaptation) to add discriminative information to such embeddings. These word embeddings were then added to a supervised WSD system by augmenting the original binary feature space with real-valued representations for all words occurring in a window of text. We evaluated our system on two general-domain lexical sample tasks, an all-words task, and also a domain-specific dataset and showed that word embeddings consistently improve the accuracy of a supervised word sense disambiguation system, across different datasets. Moreover, we observed that adding dis-

criminative information by adapting word embeddings further improves the accuracy of our word sense disambiguation system.

Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

Eneko Agirre, Oier Lopez de Lacalle, Christiane Fellbaum, Shu-Kai Hsieh, Maurizio Tesconi, Monica Monachini, Piek Vossen, and Roxanne Segers. 2010. SemEval-2010 task 17: All-words word sense disambiguation on a specific domain. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 75–80.

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

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Rie Kubota Ando. 2006. Applying alternating structure optimization to word sense disambiguation. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 77–84.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

John S Bridle. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. *Neurocomputing*, pages 227–236.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 61–72.

Yee Seng Chan and Hwee Tou Ng. 2005. Scaling up word sense disambiguation via parallel texts. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1037–1042.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 33–40.

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*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *Computing Research Repository*, abs/1207.0580.

Rob Koeling, Diana McCarthy, and John Carroll. 2005. Domain-specific sense distributions and predominant sense acquisition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 419–426.

Rémi Lebret, Joël Legrand, and Ronan Collobert. 2013. Is deep learning really necessary for word embeddings? In *Neural Information Processing Systems: Deep Learning Workshop*.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 41–48.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In *Proceedings of the 8th Conference on Computational Natural Language Learning*, pages 33–40.

Tomas Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the Eleventh Annual Conference of the International Speech Communication Association*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *In Proceedings of Workshop at International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

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*, pages 240–243.

Andriy Mnih and Geoffrey E. Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21*, pages 1081–1088.

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 40–47.

Hwee Tou Ng, Bin Wang, and Yee Seng Chan. 2003. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 455–462.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447.

Thanh Phong Pham, Hwee Tou Ng, and Wee Sun Lee. 2005. Word sense disambiguation with semi-supervised learning. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1093–1098.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1522–1531.

Tony Rose, Mark Stevenson, and Miles Whitehead. 2002. The Reuters Corpus Volume 1 – from yesterday's news to tomorrow's language resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 827–832.

Joseph Turian, Lev-Arie 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*, pages 384–394.

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.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: a wide-coverage word sense disambiguation system for free text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics System Demonstrations*, pages 78–83.

Zhi Zhong and Hwee Tou Ng. 2012. Word sense disambiguation improves information retrieval. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 273–282.