Supervised Word Sense Disambiguation Using New Features Based on Word Embeddings

Majid Fahandezi Sadi [a], Ebrahim Ansari [b,*] and Mohsen Afsharchi [a]

[a] *Department of Computer Engineering, University of Zanjan, University of Zanjan Blvd. Zanjan, Iran.*
E-mail: {mfsadi,afsharchim}@znu.ac.ir

[b] *Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences (IASBS), No. 444, Yousef Sobouti Blvd. Zanjan, Iran.*
E-mail: ansari@iasbs.ac.ir

Abstract. Supervised Word Sense Disambiguation (WSD) systems use features of the target word and its context to learn about all possible samples in an annotated dataset. Recently, word embeddings have emerged as a powerful feature in many NLP tasks. In supervised WSD, word embeddings can be used as a high-quality feature representing the context of an ambiguous word. In this paper, four improvements to existing state-of-the-art WSD methods are proposed. First, we propose a new model for assigning vector coefficients for a more precise context representation. Second, we apply a PCA dimensionality reduction process to find a better transformation of feature matrices and train a more informative model. Third, a new weighting scheme is suggested to tackle the problem of unbalanced data in standard WSD datasets and finally, a novel idea is presented to combine word embedding features extracted from different independent corpora, which uses a voting aggregator among available trained models. All of these proposals individually improve disambiguation performance on Standard English lexical sample tasks, and using the combination of all proposed ideas makes a significant improvement in the accuracy score.

Keywords: Word Sense Disambiguation, Word Embedding, Supervised Learning, Support Vector Machine

1. Introduction

Word Sense Disambiguation is a long-standing problem in computational linguistics. It is defined as the problem of finding the most probable sense of a multi-sense word in a sentence. There are mainly four approaches to solving the problem: Supervised, unsupervised, semi-supervised and knowledge-based [1]. In supervised WSD, the problem of finding the most probable sense of a word is considered as a classification task; word senses are classes and context gives some clues for training a model.

Supervised WSD systems use standard features such as Part of Speech (POS) tags, surrounding words and collocations extracted from the context of the target word, and it is assumed that they have enough information to represent the feature vector of an ambiguous word. It Makes Sense (IMS) is one of the few open-source frameworks for supervised WSD and it is so flexible that different features and classification algorithms can be employed to train models for predicting the intended sense of an ambiguous word [33].

In recent years, word embeddings have become a point of attention for NLP applications by providing invaluable information on semantic relations between words in a corpus. They have been applied successfully to many NLP tasks such as opinion mining [16], machine translation [34], named entity recogni-

[*]Corresponding author. Tel.: +98 24 3315 3380; E-mail: ansari@iasbs.ac.ir.

tion [13] , and dependency parsing [31]. Using word embeddings as a feature in a supervised WSD system was studied in several works. A recent study leveraged word embeddings as a new feature for IMS [10]. In this paper, we first introduce new coefficients and incorporate them into the feature vector generation process of word embeddings. We modify the state-of-the-art work of Iacobacci et al. [10]. Second, the effect of dimensionality reduction on word embedding based features is examined using the PCA algorithm. Third, a weighting method is used to alleviate the problem of data imbalance in available corpora. At last, a method for aggregating different word vectors from different corpora is discussed. We evaluate all of these proposed methods on two English lexical sample datasets, Senseval 2 and Senseval 3 and show that they achieve better performance compared to the previous approaches.

The main contributions of this work can be summarized as follows:
a) Using distance-based and frequency-based coefficients in building word embedding vectors for WSD tasks b) Using PCA as a pre-processing step to find a better transformation of feature matrices c) Considering the imbalanced data problem in WSD tasks by introducing a simple solution based on class weighting d) Introducing a voting strategy to exploit different word embeddings extracted from different corpora

The rest of the paper is organized as follows: Section 2 studies different WSD methods which have used word embeddings as a feature. Our proposed methods are introduced in Section 3. Section 4 includes the experiments and results, as well as comparisons with other works, and finally, Section 5 concludes the paper.

2. Related Works

Word embeddings are a group of techniques that map words from a high dimensional space, where each word is a dimension, to a much lower-dimensional space; the new one is called distributed representation. Traditionally, these word embeddings were generated using methods such as dimensionality reduction on co-occurrence matrix of words [14], or probabilistic methods [7].

Bengio et al. proposed neural language modelling and derived word embeddings using deep neural networks [3]. Mikolov et al. popularized neural word embeddings by introducing two shallow neural network models, skip-gram and CBOW. Interesting relational information can be extracted using these embeddings [18]. A deeper model to achieve word embeddings was proposed by Pennington et al. [23]. One of the main drawbacks of word embeddings is their inability to capture polysemy.

Considering word embedding applications, the existing WSD approaches can be categorized into two groups: First, works that try to modify pre-trained word embeddings or the training algorithm to achieve sense embeddings. These approaches try to solve Word Sense Induction (WSI) as a clustering problem by integrating sense embeddings into their training models. The second group of works try to use word embeddings as a feature to the supervised task of sense classification (WSD).

2.1. Sense Induction

Guo et al. proposed using translation as a tool to cluster word senses and built a monolingual sense tagged corpus. When an ambiguous word is translated to another language, to a great amount, the ambiguity is not present in the target language. Training a recurrent neural network on word clusters results in sense embeddings [8].

Another line of work deals with the training process of skip-gram to achieve sense specific word embeddings [29,21,15]. Some other works use knowledge bases or sense inventories to learn sense embeddings. Rothe and Schütze introduced Autoextend, a system which acquires word embeddings as input and derives embeddings for synsets and lexemes using WordNet [19] as an external resource [27]. Iacobacci et al. used BabelNet [20] as a sense inventory to create sense vectors for word similarity tasks [9]. Pelevina et al. introduced a method based on using ego networks to cluster word vectors and induce word senses [22].

These kinds of sense embeddings are useful for improving the performance of WSI, Part of Speech tagging, Named Entity Recognition and so on.

2.2. Sense Classification

Word embeddings application in WSD mainly consists of using them as new features in a supervised learning algorithm. Taghipour and Ng applied a modified version of word embeddings to IMS system. Their strategy for incorporating word vectors in WSD was to use the vectors of all surrounding words of the target word in a given window as new features. They improved English lexical sample and all words tasks [28]. Chen et al. introduced a knowledge-based approach to WSD using word embeddings; they built context vectors and sense vectors for each target word and ranked word senses based on two simple algorithms to measure the similarity between the context vector and sense vectors [4].

Iacobacci et al. introduced a new method for using word embeddings as features to a WSD system [10]. We modified this work, proposing four different ideas. They will be discussed in more details in the next section.

There is also a recent trend toward using neural networks to improve the performance of WSD. Context2vec is a neural network architecture which is based on word2vec [18] and generates context vectors for every target word in a corpus using Long Short-Term Memory (LSTM). Then the resulting vectors could be used in several NLP tasks including WSD. Kågebäck and Salomonsson proposed a language-independent WSD system which uses bidirectional LSTM architecture [12]. Yuan et al. proposed two methods for improving WSD tasks, the first one is an LSTM based algorithm which tries to predict a held-out word using the surrounding context words. Their second idea is using a semi-supervised approach to label more data given some labeled samples based on label propagation. However, the best performance was achieved by combining the two ideas [32]. Raganato et al. introduced a new perspective for supervised WSD in which they used neural models to disambiguate a sequence of words instead of creating a single classifier per word. They evaluated different neural models and found that sequence learning is the best performing and most consistent method based on different tasks in different languages [25]. Pesaranghader et al. used a Bidirectional LSTM to disambiguate all words in a text document without having to train a classifier per word. Their network architecture includes sense and word embedding layer and considers word order [24].

3. Proposed Methods

Here we introduce our new proposed methods regarding a supervised WSD system which uses word embeddings as features. The First two methods generate word embeddings feature vectors in an efficient way, the third one uses a weighting strategy to solve data imbalance problem and the last one introduces a voting plan between different models on different types of word embeddings.

Figure 1 shows the block diagram of the entire system. The bottom right of the figure illustrates our proposed voting mechanism. When a new sample enters to the system, three SVM classifiers trained using different embedding types, give three lists of probabilities for all of the senses of the target word, and the class with the maximum sum is selected as the correct corresponding sense.

The second and third ideas, Using PCA and weighting schema respectively, are shown on the top of the figure using dashed line rectangles. All four proposed ideas in this paper are independent of each other and could be considered as an independent extension to IMS system [33]. In our methodology, we investigated the different combinations of those proposed methods to find the best possible solution for our working system. The following sub-sections introduce each of the ideas in details.

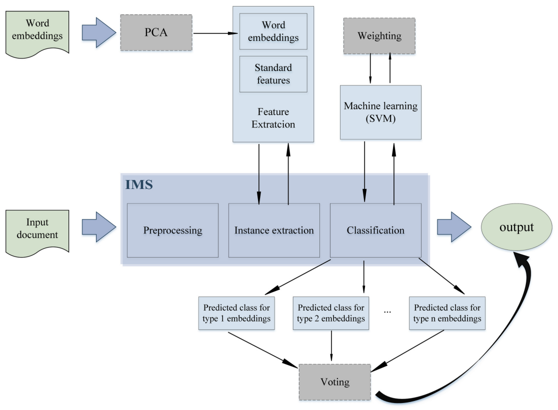

Fig. 1. The block diagram of the entire system including IMS framework and the proposed ideas which are integrated into the system. The second and third ideas, weighting system and using PCA respectively, are shown on the top of the figure and the voting scheme is shown on the bottom right of the figure divided using the dotted line shapes.

3.1. Improved Coefficient Scheme

Iacobacci et al. proposed four different strategies in order to use word embeddings as features of the supervised WSD system [10]:

1. Concatenation: in this strategy, the word vectors of the context words of the target word are concatenated together to make a big vector equal to the sizes of all vectors in the context window. Given W, as window size and D as vector dimensionality, and *I* as the index of the target word, i^{th} dimension of the feature vector, e_i, is given as:

$$e_i = \begin{cases} w_{i \bmod D,\, I-W+\lfloor \frac{i}{D} \rfloor} & \text{if } \lfloor \frac{i}{D} \rfloor < W \\ w_{i \bmod D,\, I-W+1+\lfloor \frac{i}{D} \rfloor} & \text{otherwise} \end{cases} \quad (1)$$

where w_{ij} is the weight associated with the i^{th} dimension of the vector of j^{th} word in the sentence.

2. Averaging: in this strategy a regular average over the word vector of context words is used. The average strategy calculates e_i as:

$$e_i = \sum_{\substack{j=I-W \\ j \neq I}}^{I+W} \frac{w_{ij}}{2W} \quad (2)$$

where *I* is the index of the target word and w_{ij} is the i^{th} dimension of the j^{th} word in the window.

3. Fractional Decay: in this strategy a weighted average of vectors for each context word is calculated with weights based on the distance from the target word:

$$e_i = \sum_{\substack{j=I-W \\ j \neq I}}^{I+W} w_{ij} \frac{W - |\mathrm{I} - j|}{W} \quad (3)$$

4. Exponential Decay: this strategy has achieved the best performance among the four strategies. The vector values are calculated using:

$$e_i = \sum_{\substack{j=I-W \\ j \neq I}}^{I+W} w_{ij} (1-\alpha)^{|\mathrm{I}-j|-1} \quad (4)$$

where $\alpha = 1 - 0.1^{(W-I)^{-1}}$. Here, weights exponentially decay as the distance to the target word increases.

In our experiments the Exponential Decay strategy has the best performance and accuracy among the four strategies. It shows that not only the distance from the target word is important and vectors should be weighted based on it, but also when the weighting is in the exponential form, it has the best effect on the resulting averaged vector of context.

In the proposed method, two different coefficients were used to capture more information from the context of the target word and try to generate a richer word embedding feature. The first coefficient is surrounding word distance from the target word. This distance is not a mere sentential distance but is calculated as the Euclidean distance of the target word in vector space from each of the surrounding words in its context.

For example, in the sentence "*I went to the **bank** to deposit money*", although both "went" and "money" have a sentential distance of 3 to "bank", in vector space of word embeddings, the Euclidean distance of "money" to "bank" is probably smaller than that of "went" and "bank". Our intuition behind using this coefficient is that words that are closer in terms of vector distance, i.e. words that have similar meanings, should contribute more than distant and non-similar words. A new equation to corporate word vector similarities is defined as follows:

$$e_i = \sum_{\substack{j=I-W \\ j \neq I}}^{I+W} dist_{ij} \cdot w_{ij} (1-\alpha)^{|\mathrm{I}-j|-1} \quad (5)$$

where, $dist_{ij}$ is the vector distance of word *j* from word *i*; all the other parameters remain unchanged from Eq. (4).

The second coefficient is the word frequency coefficient. Again considering the *bank* example, words such as "*I*, *to*, *the*" are very frequent words which are known as English stopwords. We did not decide to remove stopwords but weighed the feature vector of word embeddings on inverse term frequency (TF) to reduce the effect of high-frequency words on the resulting feature vectors. English Wikipedia was used for counting word occurrences and the following equation shows how to calculate weights based on word frequencies:

$$e_i = \sum_{\substack{j=I-W \\ j \neq I}}^{I+W} count_j \cdot w_{ij} (1-\alpha)^{|\mathrm{I}-j|-1} \quad (6)$$

where *count*$_j$ is the inverse of the frequency of the word *j* in English Wikipedia, and again all the other parameters remained unchanged.

3.2. *Applying Dimensionality Reduction*

Inspired by the work of Raunak, which presents a dimensionality reduction and post-processing technique to reduce the size of word embeddings [26], we performed a linear dimensionality reduction using PCA algorithm on pre-trained word embeddings to evaluate its effects on WSD performance.

Doing a PCA while holding the same dimensions as input dimensions yields a better result comparing to the real reduction of vectors. So long as the calculations after this step are not linear, the results could be biased towards a given representation over others. Therefore, the resulting dimensions of PCA were set as the input vectors dimensions. In this sense, PCA is doing a linear transformation, better distributing features in the n-dimensional space. Based on our experiments, the post-processing algorithm [26], did not improve the performance of our models. So only the best output representations of PCA are used in our work.

3.3. *New Weighting Scheme*

The sense frequency distribution of multi-sense words is not uniform, and the same unbalanced characteristic is also seen in standard WSD datasets such as Senseval 2 and Senseval 3. In supervised WSD, every sense is a class and a supervised learning algorithm should assign a class, i.e. sense, to a sample. In a supervised learning task, when the number of training samples of different classes is different, a problem arises that is known as imbalanced data [6].

Table 1 shows sense distribution of one word from Senseval 2 and one word from Senseval 3 English Lexical Sample tasks as an example of imbalanced data in WSD datasets. In an imbalanced dataset, a classifier develops a bias towards the majority class (classes) because the minority class (classes) is treated as noise. Several methods have been proposed to deal with imbalanced data problem such as undersampling, oversampling, using ensemble classifiers, and cost-sensitive methods [6]. We use a simple approach based on the latter.

Table 1

Sense frequency of two sample words from Senseval 2 (Cool) and Senseval 3 (Party) English Lexical Sample tasks

	WORD	Cool (a)	Party (n)
Number of each sense in training set	sense 1	53	148
	sense 2	25	15
	sense 3	3	16
	sense 4	8	39
	sense 5	0	17
	sense 6	1	0
	sense 7	18	0

Support Vector Machine (SVM) is a popular discriminative classifier defined by a separating hyper-plane. Given a number of points in an n-dimensional space, SVM tries to find an optimal hyper-plane which separates these data points into two classes (Although real WSD datasets have more than two classes, SVM can be generalized to support multi-label classification). A possible hyper-plane can be represented by:

$$W\prime \cdot \Phi(x + b) = 0. \tag{7}$$

where W$\prime$ is the weight vector normal to the hyper-plane and $\Phi(x)$ is the mapping function that transforms data points to a higher dimensional space. So the maximum margin hyper-plane can be found by solving the following optimization problem [30]:

$$\begin{gathered}\min(\frac{1}{2}W\prime \cdot W\prime + C^{+} \sum_{i|y_j=+1}^{l} \zeta_i + C^{-} \sum_{i|y_j=-1}^{l} \zeta_i) \\ s.t. \quad y_i(W\prime \cdot \Phi(x_i) + b) \geq 1 - \zeta_i \\ \zeta_i \geq 0, \quad i = 1, ...l\end{gathered} \tag{8}$$

where ζ_i is the slack variable for misclassified samples, such that $\sum_{i|y_j=+1}^{l} \zeta_i$ and $\sum_{i|y_j=-1}^{l} \zeta_i$ are the penalty for the total amount of training errors for positive and negative classes respectively. In case of imbalanced data, parameter C can control the amount of the penalty for weights on each class. Now C^{+} and C^{-} can be chosen in a way that reduces the effect of data imbalance. Akbani et al. argued that by setting C^{-}/C^{+} equal to the minority to majority class ratio, an optimal solution is obtained [2].

In our proposed method for handling imbalanced data in a multi-label classification task using SVM, the C parameter of each class is computed as follows:

$$C_i = \max(S)/\text{count}(i) \tag{9}$$

where S is the set of all sense counts for a word and count(i) is the number of occurrences of sense i (class i) of that word.

3.4. *Voting as a Word Embeddings Aggregation Method*

One of the main drawbacks of word embeddings is their inability to capture polysemy. For every word such as *bank*, there exists exactly one vector in the vector space. The word *bank* has 10 senses as a noun according to WordNet [19]. So the vector representation for this word is the combination of all the 10 senses. Our idea is that using different embeddings built on different corpora, or built using different algorithms, one can capture more information.

Figure 2 shows the most similar words to the word *apple* using word embeddings built on Wikipedia corpus and Figure 3 shows the most similar words to the word *apple* using word embeddings built on Google News corpus. These two figures show an important point: each corpus has its own domain, and word embeddings which are built on one corpus have different biases, in terms of word sense, to the other. This simple assumption was not considered in previous works as far as we know.

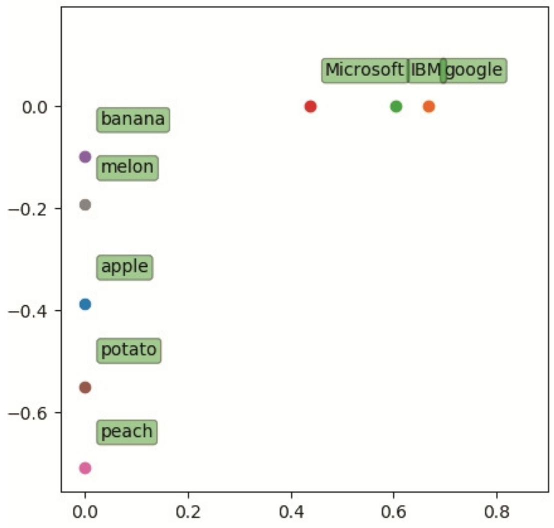

Fig. 2. The vector space of Wikipedia embeddings, where apple is near the fruit sense and far from the company sense.

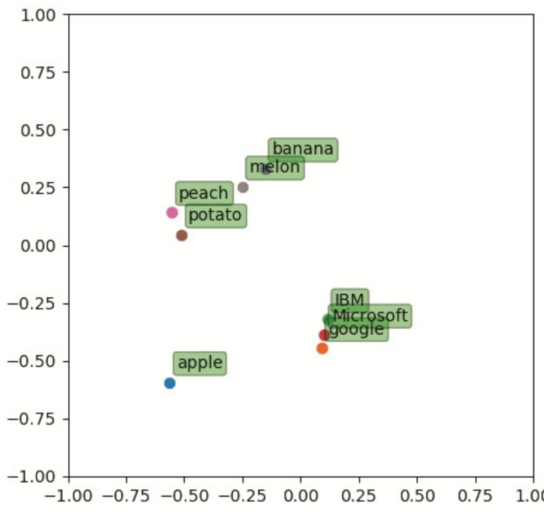

Fig. 3. The vector space of Google news embeddings, where apple is located between fruit sense and company sense.

We have selected three different kinds of word embeddings from two different corpora. The first one is the word vectors for the 2014 Wikipedia dump which were used by Iacobacci et al. [10]; it has 400 dimensions and was trained using word2vec [18]. The second word vectors were trained using fasttext algorithm[1][11] with 300 dimensions and the last one is Google News[2] word embeddings with 300 dimensions which was trained using word2vec.

Each of these word embeddings is used as a feature to the supervised learning algorithm, and the algorithm gives a probability for each sense of a polysemous word. Using the following equation, the most probable sense of a word *w* in question is chosen:

$$Sense_v = \arg\max_s f(w, s) \tag{10}$$

$$f(w, s) = \textstyle\sum_{i=1}^{n} (s_i | s_i \textit{ is the probability of sense s of word w for embedding type i.})$$

where n is the total number of embedding types. Here the sense with maximum total probability among different embedding types is chosen as the target sense of the given word *w*.

[1]https://github.com/facebookresearch/fastText/blob/masteer/pretrained-vectors.md

[2]https://code.google.com/archive/p/word2vec/

Table 2
Different used word embeddings specifications

Word Embeddings	Dimensions	Tokens
Wikipedia 2014	400	1604163
Google News	300	3000000
Fasttext	300	2519370

In case of using only one type of word embeddings, according to our experiments, Wikipedia embeddings had a better performance compared to the other two, but the voting between these three embeddings yields a better result. This shows that voting schema is a robust technique; however using more embeddings as voters not necessarily improves the result. Table 2 summarizes parameters of three word embedding types.

4. Results and Discussion

We evaluated the proposed methods on English lexical sample tasks. Senseval 2 [5] and Senseval 3 [17] challenges provide standard training and test data for English WSD tasks. Lexical sample includes training data for a number of selected words; each sample is a paragraph which contains the target word and its surrounding context. We have used IMS [33] to train a model for every word and the default classifier is a linear Support Vector Machine (SVM).

4.1. Experimental Setup

Table 3 shows the number of word types, training samples and test samples for Senseval 2 and Senseval 3 English lexical sample tasks.

In all of the proposed methods, we have used an Exponential Decay strategy [10] to generate word embeddings feature vectors. The baseline WSD system is IMS which uses SVM as the classifier. In addition to the word embeddings feature, standard WSD features, surrounding words, POS tags of surrounding words, and collocations were used too.

Table 3
Information about Senseval 2 and Senseval 3 English Lexical Sample datasets

	Senseval 2	Senseval 3
Word Types	73	57
Training Samples	8611	8022
Test Samples	4328	3944

4.1.1. Details of New Coefficients

According to our experiments, we found that the basic and original sentential distance coefficient proposed by Iacobacci et al. in Eq. (4) is effective enough. So we decided to combine our new proposed coefficients to existing coefficient proposed by Iacobacci et al. [10]. The first 400 entries of the word embedding feature vector are reserved for the original Exponential Decay strategy indicated at Iacobacci et al. and using the second 400 entries (800 in total), two models are considered to integrate the proposed coefficients into the system as follows:

1. Distance based coefficient:
   - Version 1. Combining the original coefficients and the new ones:

$$e_i = \sum_{\substack{j=I-W \\ j\neq I}}^{I+W} dist_{ij} \cdot w_{ij} (1-\alpha)^{|I-j|-1}, \tag{11}$$

     We referred to this method at our result tables as Coeff (V1).
   - Version 2. Omitting the original coefficient:

$$e_i = \sum_{\substack{j=I-W \\ j\neq I}}^{I+W} dist_{ij} \cdot w_{ij} \tag{12}$$

     We referred to this method at our result tables as Coeff (V2).
2. Word frequency based coefficient:
   - Version 1. Combining the original coefficients and the new ones:

$$e_i = \sum_{\substack{j=I-W \\ j\neq I}}^{I+W} count_j \cdot w_{ij} (1-\alpha)^{|I-j|-1}, \tag{13}$$

     We referred to this method at our result tables as Wcount (V1).
   - Version 2. Omitting the original coefficient:

$$e_i = \sum_{\substack{j=I-W \\ j\neq I}}^{I+W} count_j \cdot w_{ij} \tag{14}$$

We referred to this method at our result tables as Wcount (V2).

4.2. Results

Table 4 shows the results of Senseval 2 English lexical sample task for our proposed methods discussed in Subsections sections 3.1 to 3.3. We compared our results with the work of Iacobacci et al. [10] because it outperformed all of the previous works [27,28,4]. It can be seen that almost all of the proposed ideas are similar to or better than the baseline. In some cases, when using one of the proposed ideas individually leads to a worse result, e.g., word count coefficient or Wcount in Senseval 2, and distant coefficient or Coeff in Senseval 3 dataset, the combination idea helps us to achieve a better result.

Table 4

The result of all proposed methods on Senseval 2 English Lexical Sample task. IMSE is the baseline [10], and Coeff, Weight and PCA are our new coefficient, weighting scheme and PCA based methods, respectively. The number of all test cases is 4328. F_1: F_1 score percentage.

Method	**Correct**	**F_1**
IMSE*	3070	70.9
IMSE + Coeff (V1)	3068	70.9
IMSE + Coeff (V2)	3071	71.0
IMSE + Wcount (V1)	3064	70.8
IMSE + Wcount (V2)	3065	70.8
IMSE + Weight	3079	71.1
IMSE + PCA	3076	71.1
IMSE + PCA (400) + Coeff (V1)	3075	71.0
IMSE + PCA (400) + Coeff (V2)	**3092**	**71.4**
IMSE + PCA (400) + Wcount (V1)	3076	71.1
IMSE + PCA (400) + Wcount (V2)	3075	71.0
IMSE + PCA (400) + Weight	3084	71.3
IMSE + PCA (400) + Coeff (V1) + Weight	3088	71.3
IMSE + PCA (400) + Coeff (V2) + Weight	3086	71.3
IMSE + PCA (400) + Wcount (V1) + Weight	3086	71.3
IMSE + PCA (400) + Wcount (V2) + Weight	3085	71.3
IMSE + Coeff (V1) + Weight	3086	71.3
IMSE + Coeff (V2) + Weight	3085	71.3
IMSE + Wcount (V1) + Weight	3077	71.1
IMSE + Wcount (V2) + Weight	3078	71.1

Similarly, Table 5 shows the results for Senseval 3 English lexical sample task. In both of the tasks, the new coefficient method, weighting scheme, dimensionality reduction technique, and different combinations of these methods are individually assessed as well as combinations of all methods.

Table 5

The result of all proposed methods on Senseval 3 English Lexical Sample task. IMSE is the baseline [10], and Coeff, Weight and PCA represent our new coefficient, weighting scheme and PCA based methods, respectively. The number of all test cases is 3944. F_1: F_1 score percentage.

Method	**Correct**	**F_1**
IMSE*	2990	75.8
IMSE + Coeff (V1)	2989	75.8
IMSE + Coeff (V2)	2993	75.9
IMSE + Wcount (V1)	**2997**	**76.0**
IMSE + Wcount (V2)	2989	75.8
IMSE + Weight	2995	75.9
IMSE + PCA	2993	75.9
IMSE + PCA (400) + Coeff (V1)	2988	75.8
IMSE + PCA (400) + Coeff (V2)	2995	75.9
IMSE + PCA (400) + Wcount (V1)	2995	75.9
IMSE + PCA (400) + Wcount (V2)	2989	75.8
IMSE + PCA (400) + Weight	2989	75.8
IMSE + PCA (400) + Coeff (V1) + Weight	2991	75.8
IMSE + PCA (400) + Coeff (V2) + Weight	2995	75.9
IMSE + PCA (400) + Wcount (V1) + Weight	**2998**	**76.0**
IMSE + PCA (400) + Wcount (V2) + Weight	2989	75.8
IMSE + Coeff (V1) + Weight	2988	75.8
IMSE + Coeff (V2) + Weight	2995	75.9
IMSE + Wcount (V1) + Weight	2991	75.8
IMSE + Wcount (V2) + Weight	2991	75.8

The last method, voting scheme, was separately compared to the baseline because it is a different method and could be independently applied to every WSD system which uses word embeddings as a feature. The result of this method is shown in Table 6.

Table 6

The comparison between baseline (IMSE) and the new voting system (IMSE + Voting) on both Senseval 2 (with 4328 samples) and Senseval 3 English Lexical Sample tasks (with 3944 samples). F_1: F_1 score percentage.

	Senseval 2		**Senseval 3**	
Method	**Correct**	**F_1**	**Correct**	**F_1**
IMSE	3070	70.9	2990	75.8
IMSE + Voting	3071	71.0	3004	76.2

4.3. Analysis

Our results show that our proposed methods outperform the work of Iacobacci et al. [10] in both Senseval 2 and Senseval 3 datasets. Although almost all of the methods have an improvement over the baseline, the best performance is achieved by applying a combination of all three ideas, namely the new coefficient, weighting and using PCA, into the base system. In some cases, the combination strategy is not the best choice, e.g. combining weighting idea and PCA feature reduction in Senseval 3 does not improve the accuracy at all. As it was discussed in Section 3, the idea of voting among different systems using different word embeddings was successful in our experiments. The important point is that based on our experiments both Fasttext and Google news Embeddings had a lower F_1 score comparing to Wikipedia word embeddings when trained separately; but voting among these three– Wikipedia, Fasttext and Google news– had a better result.

Furthermore, the proposed weighting scheme, could be applied to any standard dataset whose data is not balanced enough, and since word senses naturally have a non-uniform distribution, this method may improve the result in other natural language processing tasks.

5. Conclusion

At the moment, supervised WSD approaches outperform other available approaches. Exploitation of word embeddings as a feature representation for semantic information of words in the context of an ambiguous word has recently been introduced in WSD. In this work, we introduced the following different ideas to improve WSD accuracy, and measured encouraging performance improvements on both standard WSD tasks (Senseval 1 and Senseval 2):

- Using a new coefficient scheme which is applied to a state-of-the-art supervised WSD system (IMS) which uses word embeddings as a high-quality feature vector
- Applying PCA as a dimensionality reduction technique in order to find a better transformation of word embedding feature vectors before training our supervised models
- Using a weighting system to decrease the negative effects of data imbalance in existing WSD datasets on accuracy
- A novel voting idea to aggregate word embeddings created from different corpora

All of the proposed ideas were evaluated individually on standard English lexical sample tasks and results show a consistent improvement over the baseline. Also, a combination of our ideas and voting scheme outperform the baseline and all individual F_1 scores with a score of 71.4% and 76.2% for Senseval 2 and Senseval 3 tasks respectively (compared to the baseline scores of 70.9% and 75.8%).

References

[1] Eneko Agirre and Philip Edmonds. 2007. *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science & Business Media.

[2] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. 2004. Applying support vector machines to imbalanced datasets. In *European conference on machine learning*, pages 39–50. Springer.

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

[4] Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035.

[5] Philip Edmonds and Scott Cotton. 2001. Senseval-2: overview. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5. Association for Computational Linguistics.

[6] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 2012. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484.

[7] Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. 2007. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*, 8(Oct):2265–2295.

[8] Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 497–507.

[9] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 95–105.

[10] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 897–907.

[11] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

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

[13] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

[14] Rémi Lebret and Ronan Collobert. 2013. Word emdeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*.

[15] Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? *arXiv preprint arXiv:1506.01070*.

[16] Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443.

[17] Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The senseval-3 english lexical sample task. In *Proceedings of SENSEVAL-3, the third international workshop on the evaluation of systems for the semantic analysis of text*.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[19] George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

[20] Roberto Navigli and Simone Paolo Ponzetto. 2010. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 216–225. Association for Computational Linguistics.

[21] 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*.

[22] Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. 2017. Making sense of word embeddings. *arXiv preprint arXiv:1708.03390*.

[23] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[24] Pesaranghader, A., Pesaranghader, A., Matwin, S., and Sokolova, M. (2018). One single deep bidirectional lstm network for word sense disambiguation of text data. In *Advances in Artificial Intelligence: 31st Canadian Conference on Artificial Intelligence, Canadian AI 2018, Toronto, ON, Canada, May 8–11, 2018, Proceedings 31*, pages 96–107. Springer.

[25] Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167.

[26] Vikas Raunak. 2017. Effective dimensionality reduction for word embeddings. *arXiv preprint arXiv:1708.03629*.

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

[28] Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323.

[29] Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 151–160.

[30] Konstantinos Veropoulos, Colin Campbell, Nello Cristianini, et al. 1999. Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI*, volume 55, page 60.

[31] Xiang Yu and Ngoc Thang Vu. 2017. Character composition model with convolutional neural networks for dependency parsing on morphologically rich languages. *arXiv preprint arXiv:1705.10814*.

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

[33] 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*, pages 78–83. Association for Computational Linguistics.

[34] Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.