Journal of Biomedical Informatics 73 (2017) 137–147

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Word embeddings and recurrent neural networks based on Long-Short Term Memory nodes in supervised biomedical word sense disambiguation

Antonio Jimeno Yepes

IBM Research Australia, Melbourne, VIC, Australia

ARTICLE INFO

Article history:
Received 2 April 2016
Revised 27 July 2017
Accepted 2 August 2017
Available online 7 August 2017

Keywords:
Word sense disambiguation
Word embeddings
Recurrent neural networks
Biomedical domain

ABSTRACT

Word sense disambiguation helps identifying the proper sense of ambiguous words in text. With large terminologies such as the UMLS Metathesaurus ambiguities appear and highly effective disambiguation methods are required. Supervised learning algorithm methods are used as one of the approaches to perform disambiguation. Features extracted from the context of an ambiguous word are used to identify the proper sense of such a word. The type of features have an impact on machine learning methods, thus affect disambiguation performance. In this work, we have evaluated several types of features derived from the context of the ambiguous word and we have explored as well more global features derived from MEDLINE using word embeddings. Results show that word embeddings improve the performance of more traditional features and allow as well using recurrent neural network classifiers based on Long-Short Term Memory (LSTM) nodes. The combination of unigrams and word embeddings with an SVM sets a new state of the art performance with a macro accuracy of 95.97 in the MSH WSD data set.

1. Introduction

The amount of biomedical text published is growing exponentially and researchers are finding it increasingly difficult to find relevant information. The automatic processing of biomedical articles can help with this problem by identifying biomedical entities (such as genes, diseases, drugs), and the relations between them. This information can be extracted from text and used for applications such as summarization, data mining and intelligent search. However, identifying biomedical entities and relations in text is a complex and challenging task.

One difficulty, addressed by this research, is the problem of lexical ambiguity. Lexical ambiguity is the presence of two or more possible meanings within a single term or phrase. For example, determining whether the term *bass* is referring to a *fish* or *instrument* given the context in which the term is used. Disambiguation is useful in concept mapping algorithms and tools relying on dictionary look up, such as MetaMap [3].

The goal of word sense disambiguation (WSD) is to automatically predict the most likely sense of an ambiguous word. For instance, the word *cold* could refer to the temperature or an infection depending on the context in which it is used. A WSD algorithm would predict the most appropriate sense given the context of the ambiguous word.

There are several approaches being used for WSD which range from supervised approaches (which rely on examples of use of each ambiguous word in context to train a learning algorithm) to knowledge-engineering approaches (which rely on a sense catalog such as a dictionary).

In this work, we explore the use of word embeddings as candidate representations for the WSD problem. We show that unigram representation is a strong baseline using Support Vector Machines as the machine learning algorithm, but that neural network word embeddings improve theses baseline results. We explore as well the different parameters used in the generation of word embeddings.

Furthermore, a combination of word embeddings and unigam features with SVM set a new state of the art disambiguation in macro accuracy of 95.97 in the MSH WSD data set.

2. Related work

WSD algorithms utilize the context in which a term is used to identify the appropriate sense of a lexical ambiguity. Existing disambiguation algorithms to resolve ambiguity can be divided into three groups: unsupervised [31,6,7], supervised [42,35], and

E-mail address: antonio.jimeno@au1.ibm.com

http://dx.doi.org/10.1016/j.jbi.2017.08.001
1532-0464/© 2017 Elsevier Inc. All rights reserved.

knowledge-based [30,2,27,19] algorithms. Unsupervised algorithms typically use clustering techniques to divide occurrences of an ambiguous word into groups that are later associated with their possible sense and might help identify new senses [23]. Supervised algorithms use machine learning techniques to assign concepts to instances containing the ambiguous word, thus these methods require examples of use of the different senses of the ambiguous words for model training. Knowledge-based algorithms do not require a corpus containing examples of the ambiguity but rather use information from an external knowledge source such as a taxonomy or dictionary.

In this work, we focus on supervised learning algorithms with the intention of exploring higher order features. Even though developing data sets for supervised methods is expensive, we believe that the insights learned by exploring features with supervised methods can be beneficial for other kinds of methods.

As in many supervised learning tasks, representation of the problem is relevant to the performance of the task [18], i.e. transforming text into features to be used by machine learning algorithms. There are several feature sets used in previous work [1,29], this includes local features, topical features and syntactic dependencies.

Stevenson et al. [35] have shown that using linguistic features in combination with meta-data of the published articles (e.g. MeSH® headings) improve disambiguation performance, even though manually annotated meta-data features cannot be assumed to be always available. McInnes et al. [26] used the annotation provided by MetaMap to automatically assign UMLS® concept identifiers. Overall, using additional features to unigrams improves the WSD performance.

The features engineered in previous work on biomedical WSD have focused on local features derived from the context of the ambiguous word or meta-data of the citation. We would like to take a step further and consider higher order features with supervised learning algorithms. These features can be seen as a more global representation, compared to locally derived features.

In Natural Language Processing, there are new algorithms developed based on neural networks that are capable of learning a representation of the bag-of-word features into a continuous bag-of-words representation [4]. This continuous bag-of-words representation can place terms with similar meaning closer and typically tend to work with lower dimensionality [28], e.g. 100 dimensions. Furthermore, this representation is more compact compared to the sparse bag-of-words.

Word embeddings has been previously used in WSD. Chen et al. [8] and Rothe and Schütze [33] show approaches using word embeddings in knowledge-based approaches obtaining state-of-the art performance. Taghipour and Ng [38] and Sugawara et al. [36] recently experimented with several features with SVM in supervised WSD improving more traditional features. In our work, we explore word embeddings in biomedical word sense disambiguation.

Word embeddings have been used with recurrent neural networks. Some advantages of using word embeddings is the lower dimensionality compared to bag-of-words and that words close in meaning are closer in the word embedding space. Very recent work still under preprint on using a special kind of recurrent network named LSTM (Long Short Term Memory) for WSD is recently being made available [40] and with bidirectional LSTM [21], improving over more traditional supervised learning methods.

3. Methods

We have compared several feature types, which are explained in more detail in this section. These feature types range from standard unigram and bigram features to more sophisticated ones based on word embeddings.

3.1. Evaluation data sets

We evaluate the different feature sets using the MSH WSD data set [19] and the NLM WSD data set [39]. Both data sets are available from https://wsd.nlm.nih.gov.

3.1.1. MSH WSD data set

MSH WSD was automatically developed by first screening the UMLS Metathesaurus to identify ambiguous terms whose possible senses consist of two or more MeSH headings. Each ambiguous term and its corresponding MeSH heading is used to extract MEDLINE citations where the term and only one of the MeSH headings co-occur, based on the MeSH headings assigned to the citation. The term found in the MEDLINE citation is automatically assigned the UMLS CUI from the 2009AB UMLS version linked to the MeSH heading.

MSH WSD contains 106 ambiguous abbreviations, 88 ambiguous terms and 9 which are a combination of both, for a total of 203 ambiguous entities. For each one of these entities, the data set contains a maximum of 100 instances per sense obtained from the 2010 MEDLINE baseline. Each target word contains approximately 187 instances, has 2.08 possible concepts and has a 54.5% majority sense. Previous supervised WSD results using Naïve Bayes showed a macro average accuracy over 93% [19].

3.1.2. NLM WSD data set

The NLM WSD data set [39] has been used to conduct the experiments. This set contains 50 ambiguous terms that have been manually annotated with a sense number. Each sense number has been related to UMLS semantic types, thus originally no UMLS concept identifiers were assigned to the senses. 100 manually disambiguated cases are provided for each term. In case no UMLS concept is appropriate, *None of the above* has been assigned.

The selection of the 50 ambiguous words was based on an ambiguity study of 409,337 citations added to the database in 1998. MetaMap was used to annotate UMLS concepts in the titles and abstracts based on the 1999 version of the UMLS. 50 highly frequent ambiguous strings were selected for inclusion in the test collection. Out of 4,051,445 ambiguous cases found in these citations, 552,153 cases are represented by these 50 terms. This means that this data set focuses on highly frequent cases.

We have considered the same setup as [19] and discarded the *None of the above* category. Since the ambiguous term *association* has been assigned entirely to *None of the above*, it has been discarded. Furthermore, there some ambiguous words in which only one of the senses was annotated, thus it is not interesting to test machine learning methods on those words. These words are: *depression*, *determination*, *fit*, *fluid*, *frequency*, *pressure*, *resistance* and *scale*. This means that we will present results for 41 out of the 50 ambiguous terms. Using the maximum frequency sense for each ambiguous word, the macro accuracy is 82.63 and the micro accuracy is 82.31.

3.2. Text based features

Citations text was extracted from the title and abstract fields. Further processing was done to the text that included lower casing, tokenization using a custom regular expression and stemming using Porter stemmer. Unigrams and bigrams were extracted from the text and experiments with bigrams were run in combination with unigrams.

Text was processed as well to add semantic annotations in addition to local features. UMLS concept identifiers (CUIs) [26] were extracted from the MEDLINE® Baseline (http://ii.nlm.nih.gov/MMBaseline/index.shtml), which is available with a version annotated with the MetaMap tool [3]. In this case, the context of the ambiguous word is represented by a bag-of-concepts instead of a bag-of-words. Another representation derived from the conceptual representation is based on UMLS Semantic Types, which is obtained from the concept annotation since UMLS concepts are assigned one or more semantic type from the UMLS Semantic Network. We have not considered meta-data since no assumption about its availability can be made.

In addition, we have considered as well two sets of features previously used in [42]. The first one is the part-of-speech (POS), thus a syntactic feature, of the three words before and after the ambiguous words. These words need to happen in the same sentence or a null value is used. The POS has been obtained from the MedPost/SKR POS tagger available from MetaMap [34]. The second one is a set of 11 local collocations features including: $C_{2,2}, C_{1,1}, C_{1,1}, C_{2,2}, C_{2,1}, C_{1,1}, C_{1,2}, C_{3,1}, C_{2,1}, C_{1,2}$, and $C_{1,3}$, where $C_{i,j}$ refers to an ordered sequence of words (n-grams) in the same sentence as the ambiguous word. Offsets i and j denote the starting and ending positions of the sequence relative to the ambiguous word. A negative or positive offset refers to a word to the left or right of the ambiguous word respectively.

Table 1 shows example features for the ambiguous word *nutrition* from the citation with PubMed identifier 9336574.

3.3. Word embeddings

Word embedding approaches transform the bag-of-words representation typically used in Natural Language Processing to a continuous space representation. There are some advantages to this continuous space since the dimensionality is largely reduced and words closer in meaning are close in this new continuous space. Existing applications to generate these embeddings based on neural networks include word2vec [28] (https://code.google.com/p/word2vec) and glove (http://nlp.stanford.edu/projects/glove).

We have used word2vec, which offers two possible ways to generate the word embeddings. The first one is called CBOW (continuous bag-of-words). The second one is skip-gram. In this work, we have used the CBOW approach, which exemplified in Fig. 1. In this approach, a neural network is trained to predict a word $W(t)$ given the words in the context in a supervised method.

We have experimented with CBOW word2vec vectors of several dimensions (100–500) and the window from which the terms are used to build the embeddings (5–150).

3.3.1. Generation of word embeddings

2014 MEDLINE is the corpus used to generate the word embeddings, which contains over 22M citations. From this corpus, we removed the citations that appear in the disambiguation data set used in the experiments, presented later in this section.

3.3.2. Aggregation of word embeddings

After the word embeddings are generated, for each word in the dictionary a vector in an n-dimensional space is available using a look up function. Prior to using the vectors in a machine learning method, the vectors from each individual word need to be combined. We have evaluated the following two methods, described as well in Fig. 2. The whole citation text has been considered for disambiguation, thus different disambiguation context length are considered that might support the use of an average method versus one based on the sum of the vectors.

Table 1
Example features for ambiguous word *nutrition* from citation with PubMed identifier 9336574.

Feature	Value
Unigrams text snippet	Charting by exception: . . .Documentation varies with the level of nutrition care.
Context	. . .risk or with *nutrition* education needs are . . .
Context features	or (−2), with (−1), education (+1), needs (+2), or (−2) with (−1), with (−1) education (+1), education (+1) needs (+2), risk (−3) or (−2) with (−1), or (−2) with (−1) education (+1), with (−1) education (+1) needs (+2), education (+1) needs (+2) are (+3)
POS	noun (−3), prep (−1), conj (−2), noun (+1), noun (+2), aux (+3)
Concepts (UMLS CUI list)	C0684240, C1554961, C1705847, C0037633, . . ., C0525069
Semantic types	inpr, idcn, cnce, sbst, bodm, . . ., orgf, orga
Word embedding	−82.9220,105.5030, . . ., −37.6584 (150 dimensions)

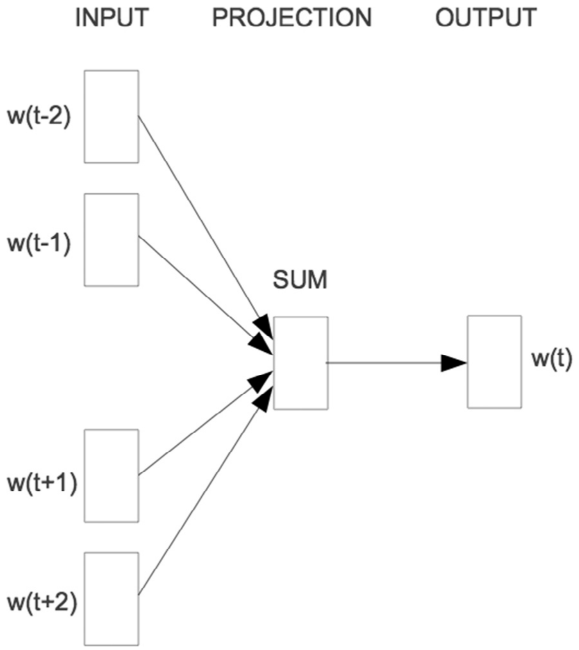

Fig. 1. Continuous bag-of-words estimation diagram [28].

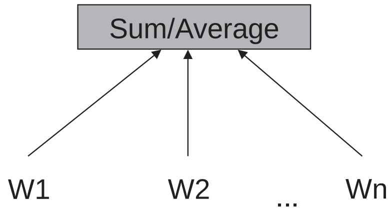

Fig. 2. Aggregation of continuous bag-of-words representation vectors.

- Sum the vectors of the words in the context of the ambiguous word. The dimensionality of this sum is the same as the vectors generated by *word2vec*. The disambiguation context is the abstract in which the ambiguous word appears, thus it is affected by the context size.

- Average the vectors of the words in the context of the ambiguous word. The dimensionality of the average is the same as the vectors generated by *word2vec*. This aggregation method accounts for different context sizes.

Table 1 shows examples of features for the ambiguous word *nutrition* from the citation with PubMed identifier 9336574.

3.4. Supervised learning algorithms

The supervised learning algorithms considered in this work are linear Support Vector Machines (SVM) based on SMO (Sequential Minimal Optimization) [32] using a linear kernel and feature normalization and Naïve Bayes (NB) [20], which are typically considered for this task. We have used the implementation provided by WEKA [15] of these algorithms for our experiments. In addition, we have considered as well k-nearest neighbors (KNN) using cosine similarity on normalized features. 1, 3 and 5k-nearest neighbors have been considered and results are shown for $k = 5$, which had the best performance.

For each ambiguous word, a classifier is trained to recognize each one of the possible senses of that word.

3.5. Long Short Term Memory

In addition to non-deep-network learning algorithms, we have used the word embeddings to train a neural network based on a Long Short Term Memory (LSTM) unit [16]. As in the case of non-deep-network methods, one LSTM based classifier is trained per ambiguous word. LSTM is a recurrent network that does not suffer from the *vanishing gradient* [5] problem and has been used in Natural Language Processing tasks [41,37].

LSTM units introduce mechanisms to avoid the *vanishing gradient* problem using, for a given time t, an input gate i_t, an output gate o_t, a forget gate f_t and a cell c_t. The weights for these three gates and memory cell that are trained using backpropagation using training data. The input to the LSTM cell is the vector x_t and the hidden output is h_t. The capability of LSTM of effectively dealing with long dependencies, e.g. syntactic dependencies, which might be useful to perform text analytics tasks such as disambiguation.

We follow the definition of LSTM unit introduced in [14], which follows the diagram in Fig. 3. Eqs. (1)–(5) show how the values in different LSTM components get calculated. Weights matrices W have subscripts that indicate the components being related. For instance W_{hi} is the weight matrix between the hidden output and the input gate.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (4)$$

$$h_t = o_t tanh(c_t) \quad (5)$$

The schema of the network is shown in Fig. 4 and offers a different approach to combine the word embeddings that take into account the document structure.

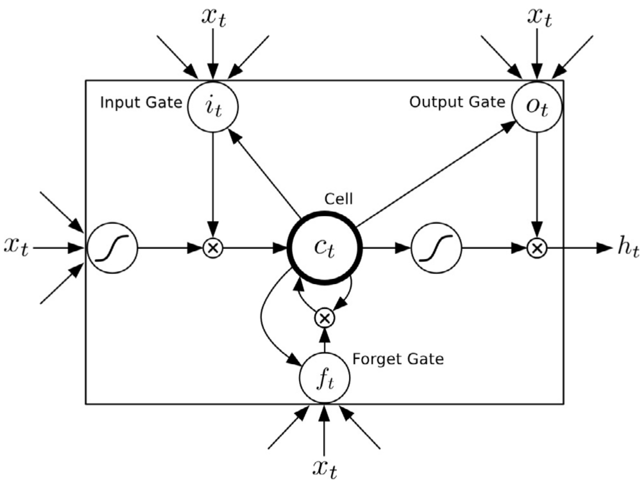

Fig. 3. LSTM memory cell unit diagram [14].

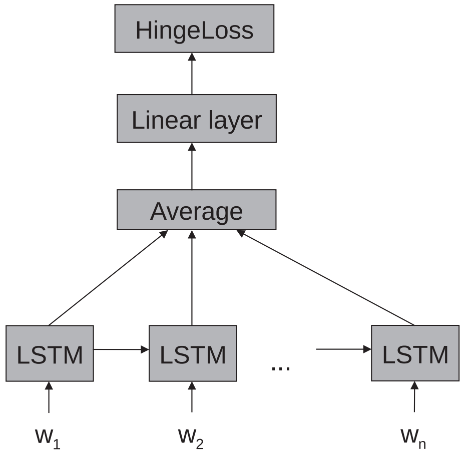

Fig. 4. LSTM layout.

The size of the LSTM memory cell and the input, output and forget gates have been set as the size of the input vector defined by the word embedding size. The output h_t of the LSTM for each word in the context of the ambiguous word is averaged and a linear layer is trained to make a decision on the averaged vector, the size of the output layer is the same as the number of senses of the ambiguous word. In the final layer, a multi-class classification Hinge loss has been used. This network structure is similar to [41], which was used in text categorization, the final layer configuration being the biggest difference.

LSTM has been implemented using Torch [10] and it has been trained using AdaGrad [12]. Learning rate has been set to 0.01 and learning rate decay to 0.01.

There are many parameters in each LSTM configuration, which may suffer from limited training data. On the other hand, there is a large quantity of unlabeled data that can be used to pre-train the recurrent neural network model. To do so, we have followed a sequence to sequence autoencoder method [11,25]. The pre-trained LSTM weights were used as initial weights of the LSTM in the supervised training instead of a random initialization. The results show a small improvement (e.g. 94.68 vs 94.64 macro accuracy for the LSTM S100 in Table 4), this difference was not significant.

4. Results

The features presented in the methods section have been evaluated using the MSH WSD, generating different feature sets. Based on these feature sets, Naïve Bayes, KNN and Support Vector Machines have been the machine learning algorithms selected to be trained for WSD. Performance results for each feature set are presented and compared. This section is divided in several sections: in the first one bag-of-word features are evaluated, then the word embeddings and finally the recurrent network based on LSTM, then several feature combination experiments are presented followed by a comparison of accuracy per ambiguous word on selected experiments. Selected features from each section have been combined to evaluate feature combination.

All experiments have been done using 10-fold cross-validation. Statistical significance has been determined using a randomization version of the paired sample t-test [9]. Accuracy has been used as the evaluation measure. Macro and micro average have been used to aggregate the performance the ambiguous words in both data sets. Confidence intervals 95 percent around the mean $(\mu \pm 1.96(\frac{\sigma}{\sqrt{N}}))$ for macro-averages are shown in selected cases in which performance of the compared methods might be close.

4.1. Text based features

Table 2 shows the results of training non-deep-network machine learning methods on features extracted from processing the citation text for WSD using the MSH WSD set. Unigrams performance is quite competitive and in the case of Naïve Bayes and KNN, bigrams performance differences significantly improve the performance of unigrams. Features such as concepts or semantic types have lower performance compared to unigrams and bigrams.

We show as well the performance when all features are combined (All). Since the performance of *POS* and *Collocations* is much lower compared to the rest of features, we show as well *All-Collocations-POS*. This feature combination improves the performance of unigrams and bigrams.

In previous work on the MeSH WSD data set, NB has been the only machine learning method used [19]. Results with SVM show that the machine learning method used affects as well the accuracy with the same feature set. KNN did not perform as well compared to the other methods.

Table 3 shows the results for the NLM WSD set. Unigrams is as well a strong feature representation, on the other hand combining different feature sets do not improve the performance over unigrams. NB is significantly better than SVM, which is another difference compared to the MSH WSD data set. All the results are better compared to the maximum frequency sense baseline (macro average: 82.63 and micro average: 82.31).

4.2. Word embeddings

In the Methods section, generation of word embedding vectors was presented. The parameters used to generate these vectors and their aggregation are used to decide the experiments to be done and are enumerated below. Each parameter configuration has been used to train a Naïve Bayes, KNN and SVM classifier.

- Size of vectors generated by word2vec: 100, 150, 300 and 500.
- Window defining how many context words are being used values are: 5, 50 and 150.
- Aggregation method: either *sum* of the vectors or their *average* is used.

Results for the different aggregations are presented as supplementary material. Averaging seems to provide better performance, with SVM obtaining better performance compared to previously published results on the MSH WSD set. Large vector size and large window seem to boost accuracy for the MSH WSD set while smaller vector size and mid window size seems to perform better for the NLM WSD set.

Naïve Bayes performance is below state of the art results using word embeddings. Weka uses the method presented in [20] for Naïve Bayes, which assumes that numerical attributes are generated by a Gaussian distribution. Loss in performance might indicate that the attributes in this new space do not follow a Gaussian distribution.

Table 2
Macro and micro average disambiguation results MSH WSD set. Best results per feature are shown in bold. Best overall result are shown underlined.

Features/ML	NB	SVM	KNN5
Unigrams	93.07/92.85	**93.90/93.77**	92.40/92.07
Bigrams	93.87/93.76	**93.94/93.86**	92.91/92.57
POS	62.16/60.89	**62.73/61.18**	49.41/49.17
Collocations	74.59/73.40	**77.34/76.24**	52.10/52.13
Concepts	**91.58/91.19**	91.18/90.93	88.91/88.53
Semantic Types	**85.89/85.27**	84.82/84.01	84.02/83.29
All	92.84/92.51	**93.78/93.57**	91.92/91.48
All-Collocations-POS	93.41/93.16	**94.36/94.19**	92.45/92.05

Table 3
Macro and micro average disambiguation results NLM WSD. Best results per feature are shown in bold. Best overall result are shown underlined.

Features/ML	NB	SVM	KNN5
Unigrams	**88.61/88.73**	87.87/88.00	87.53/87.69
Bigrams	86.57/86.81	86.35/86.26	**87.55/87.54**
POS	79.06/79.09	82.72/82.66	**82.95/82.75**
Collocations	85.08/84.90	**85.58/85.51**	83.68/83.30
Concepts	**88.06/88.03**	86.76/86.67	86.63/86.61
Semantic Types	**86.90/86.76**	86.12/86.15	86.12/85.97
All	87.50/87.74	87.40/87.39	**87.91/87.86**

4.3. Recurrent network

The LSTM network has been evaluated using vector size 100 and 500 with window 50 in the word embedding generation for the MSH WSD set and for vector size 150 and 500 with window 50 for the NLM WSD set. 10-fold cross validation has been used to obtain the results for each one of the terms.

Table 4 shows the result for the two set of vectors on the MSH WSD set. The 500 vector size has the best performance.

Table 5 shows the result for the two set of vectors on the NLM WSD set. Using word embeddings significantly improve over using unigrams. LSTM shows some improvement in macro averaging. which is not significant compared to SVM and word embeddings.

4.4. Feature combination results

We have evaluated several feature sets in this study. Table 6 shows the performance of an SVM classifier in different combinations of these features with word embeddings. As shown above word embedding features and unigrams show a significant difference in performance.

Since unigrams and word embeddings features have their own strengths depending on the ambiguous word, we have combined them with the expectation that the learning algorithm identifies the more relevant features for each ambiguous word during training [13]. The selected word embeddings used in this combination has been generated window size 50 and vector size 500 and average aggregation for the MSH WSD set and window size 50 and vector size 150 for the NLM WSD set.

On the MSH WSD set, the accuracy obtained using SVM using this combination is 95.97/95.80. The difference of results is statistically significant ($p < 0.0001$) when compared to any other evaluated method, except for WE + ST + Concepts + Unigrams (95.95/95.80; $p < 0.48, 0.0117 \pm 0.4525$). The combination of local features derived from the context of the ambiguous word and glo-

Table 4
Macro and micro average LSTM results compared to SVM unigram and bigrams and word embeddings MSH WSD set.

Configuration	Macro accuracy	Micro accuracy
SVM Unigrams	93.90	93.94
SVM Bigrams	93.94	93.81
SVM All-Collocations-POS	94.36	94.19
SVM WE S100 W150	94.50	94.31
LSTM S100 W150	94.64	94.58
SVM WE S500 W50/ S300 W150	94.64	94.49
LSTM S500 W50	**94.87**	**94.78**

Numbers in bold indicate the best accuracy for a group of results.

Table 5
Macro and micro average LSTM results compared to SVM unigram and bigrams and word embeddings NLM WSD set.

Configuration	Macro accuracy	Micro accuracy
SVM Unigrams	87.87	88.00
NB Unigrams	88.61	88.73
SVM WE S150 W50	90.58	**90.42**
LSTM S150 W50	90.63	90.02
SVM WE S500 W50	89.79	89.63
LSTM S500 W50	**90.64**	90.19

Numbers in bold indicate the best accuracy for a group of results.

Table 6
Macro and micro average feature combination study of different feature combinations including word embeddings MSH WSD. The learning algorithm is SVM and the word embedding configurations use 500 dimensions (S) and context window (W) of 50.

Features	Macro accuracy	Micro accuracy
WE + Unigrams	**95.97**	**95.80**
WE + Bigrams	95.56	95.40
WE + Concepts	95.09	94.92
WE + Semantic Types	93.95	93.69
WE + POS	93.78	93.50
WE + Collocations	94.55	94.33
WE + All	95.00	94.78
WE + ST + Concepts + Unigrams	95.95	**95.80**
WE + All-Collocations-POS	95.82	95.65

Numbers in bold indicate the best accuracy for a group of results.

Table 7
Macro and micro average feature combination study of different feature combinations including word embeddings NLM WSD.

Features	NB	SVM
WE + Unigrams	88.79/88.91	88.69/88.85
WE + Bigrams	87.50/87.54	86.86/86.84
WE + Concepts	88.73/88.76	87.64/87.57
WE + Semantic Types	87.45/87.37	88.31/88.21
WE + POS	86.76/86.58	88.85/88.93
WE + Collocations	87.22/87.08	**89.91**/**89.89**
WE + All	87.90/87.89	87.40/87.39

Numbers in bold indicate the best accuracy for a group of results.

bal features, provides a significant boost and sets a new performance on the MSH WSD set. Similar behavior has been observed by [36].

Other feature combinations either have a similar performance (e.g. when combining all of them as in *WE + ALL*) or show a significant lower difference in performance when using semantic types (e.g. *WE + Semantic Types*), which had already shown lower performance when used alone.

On the NLM WSD set, we find that the feature combination of unigrams and word embeddings and concepts and word embeddings improve slightly on using unigrams alone, which is not significant ($p < 0.30, 0.1795 \pm 0.6646$). On the other hand, the performance of the other combinations is lower compared to using unigrams alone with NB (see Table 7).

5. Discussion

We show that features types investigated in our work derived from MEDLINE, using word embeddings, in combination with non-deep-network machine learning algorithms improve results obtained with unigrams.

Averaging of word embeddings with SVM improves WSD performance compared to more traditional features. The improvement difference is statistically significant ($p < 0.03$) when the window size is larger or equal than 50 and vector size is larger than 100. Summing word embedding vectors seems to decrease performance and might be due to the effect of longer citations, the disambiguation context in this work is defined as the whole citation text. KNN results sit in between SVM and Naïve Bayes and performance using word embeddings for number of neighbors > 1 is just above using unigrams for the MSH WSD set.

As shown in results in supplementary material, typically a larger window and vector size will improve WSD performance when using non-deep-network learning methods.

Results in Tables 2 and 3 show differences in performance of classifiers. It would be interesting to further explore additional classifier parameters, e.g. using cross-validation, instead of using default parameters (e.g. C parameter for SVM).

Results show that LSTM improves the performance of non-deep-network learning algorithms when using only word embeddings. In the case of the MSH WSD set, the difference in performance is statistically significant with respect to other methods ($p < 0.005$; ci with SVM S500 W50: average difference 0.4024 ± 0.3130). Even though the differences of the best LSTM configurations is not statistically significant ($p < 0.17$; average difference 0.12445 ± 0.2506). Despite being the differences similar between the LSTM and non-deep-network methods, the accuracies per ambiguous word show less variation in the LSTM based methods.

In the NLM WSD set, LSTM improves the performance of non-deep network algorithms but the difference in performance is not significant. This may be because word embeddings already contain information for predicting a word given the context and might be seen as a kind of pre-training.

We have observed that LSTM performed worst compared to other methods when a significantly smaller number of examples are provided in the MSH WSD set (c.f. scatter plots in Figs. 5 and 6). For instance, *PAC* has only 46 and 16 examples of each one of the two senses and in the case of *hemlock*, the number of examples is 57 and 20 respectively. LSTM needs to learn a larger number of parameters, around 81,002 with word embedding vector size 100 and 2,005,002 with vector size 500. If enough examples are provided, LSTM could potentially improve other methods.

Word embedding based methods seem to improve state of the art methods when word embedding allow a better distinction of senses, as in *nursing* (profession versus breast feeding) and *labor* (childbirth versus work). On the other hand, words like *Ca*, *digestive* or *blood pressure*, in which the meanings are close, word embedding performs below state of the art methods. In these cases, a word seems to be the discriminative clue to a proper disambiguation.

We have grouped ambiguous terms according to several criteria. Table 8 shows the performance by groups of ambiguous words with a defined group of senses. Results are shown for words with 2 senses (189 words) and 3 senses (12 words). Ambiguous words

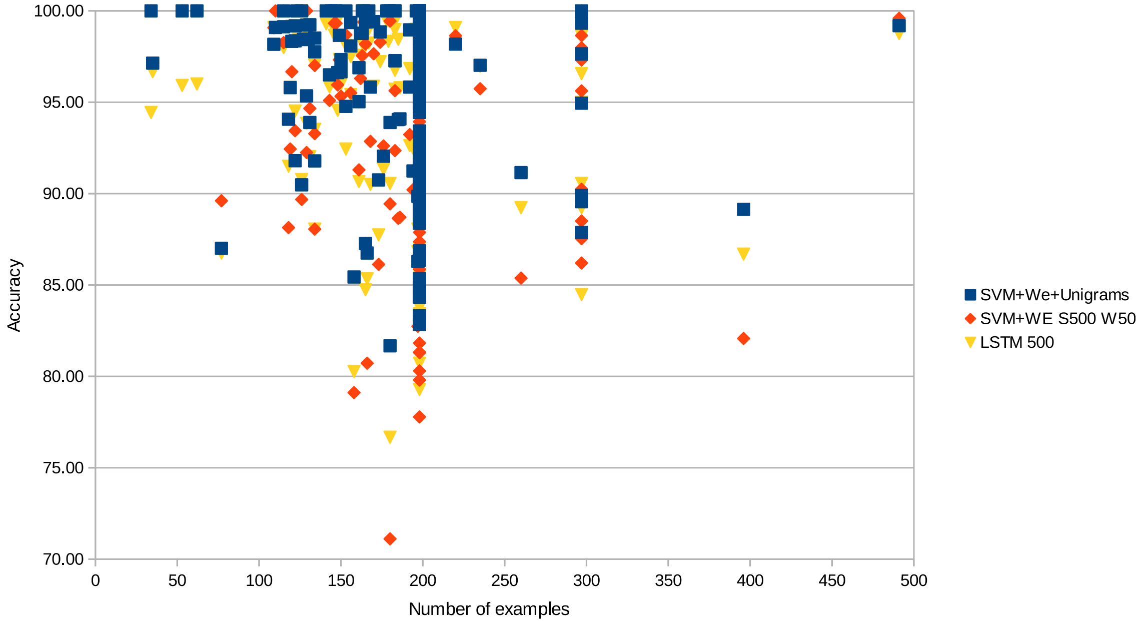

Fig. 5. MSH WSD set difference in accuracy per ambiguous word between the combination of word embeddings with unigrams (WE + Unigrams in Table 6) versus just using SVM and unigrams (Table 2) sorted in descending order.

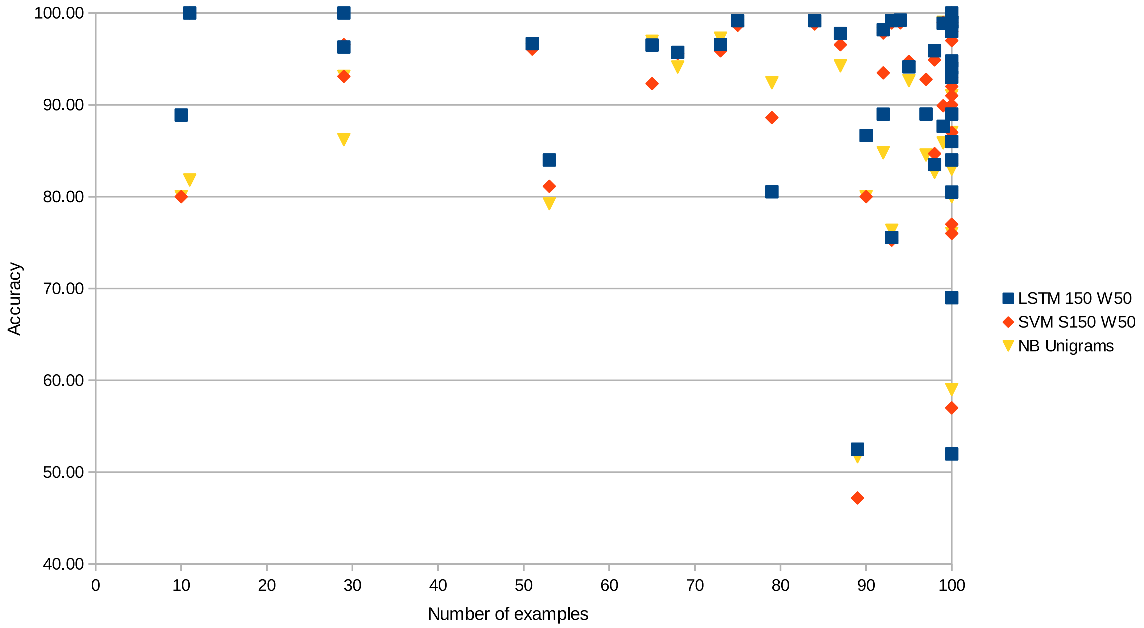

Fig. 6. MSH WSD set difference in accuracy per ambiguous word between the combination of word embeddings with unigrams (WE + Unigrams in Table 6) versus just using SVM and unigrams (Table 2) sorted in descending order.

Table 8
Macro average results for ambiguous words grouped by number of senses for the MSH WSD set.

Method	2 senses	3 senses
SVM Unigrams	94.13	93.59
SVM WE S500 W50	94.75	93.58
LSTM WE S500 W50	95.00	94.06
SVM WE S500 W50 + Unigrams	**96.03**	**95.22**

Numbers in bold indicate the best accuracy for a group of results.

with 4 and 5 senses appear only one in the data set. Macro average shows that words with 2 senses are easier to disambiguate and that words with 3 senses are slightly more complicated. Methods relying solely on word embeddings as features seem to have a larger drop in performance between 2 word senses and 3 word senses.

The NLM WSD data set has 34 ambiguous words annotated with 2 senses and 6 with 3 senses. The word *cold* is the only word with 4 senses annotated. Table 9 shows the results of ambiguous words grouped by number of senses. We find that words with 2 senses are typically disambiguated with higher accuracy, while 3 senses seem to be disambiguated with lower accuracy. There is a top of 100 examples for each ambiguous word, so if 3 senses appear, there is less training data per ambiguous word sense.

Table 9
Macro average results for ambiguous words grouped by number of senses for the NLM WSD set.

Method	2 senses	3 senses
NB unigrams	90.87	75.15
SVM Unigrams	90.03	74.81
SVM WE S150 W50	91.25	73.68
LSTM WE S150 W50	**93.40**	74.37
LSTM WE S500 W50	93.29	**75.20**

Numbers in bold indicate the best accuracy for a group of results.

Table 10
Macro average by ambiguous word type (Term (T), Abbreviation (A), Term-Abbreviation (TA)) for the MSH WSD set.

Method	T	A	AT
SVM Unigrams	90.23	97.26	94.55
SVM WE S500 W50	90.92	97.58	93.69
LSTM WE S500 W50	92.04	97.34	94.64
SVM WE S500 W50 + Unigrams	**93.07**	**98.32**	**96.50**

Numbers in bold indicate the best accuracy for a group of results.

As defined in [19], the ambiguous words in MSH WSD can be divided into terms (T), abbreviations (A) and words that might act as both (AT). Table 10 shows the macro average performance on these sets of words. Terms (T) are the most difficult to disambiguate, while abbreviations seem to be the easiest set. SVM with word embeddings and unigrams performs the best on all the categories. LSTM seems to be better for terms compared to SVM when unigrams and word embeddings are used separately. This different seems to be less clear for abbreviations and ATs. SVM with word embeddings seems to perform less well for the AT group in comparison to the performance in other categories with other methods.

5.1. Per ambiguous word accuracy differences

We have further examined the difference in performance for the MSH WSD feature sets and LSTM. Figs. 7–9 show the difference in accuracy per ambiguous term considered in this work. In most cases, the outcome of the combination improves the results obtained by either using unigrams and SVM (Fig. 7), average word embeddings with vectors size 500 and window 50 (Fig. 8) and LSTM 500 with vector size 500 and window 50 (Fig. 9). The differences in favor of the combination are more prominent when compared to unigram results with terms like *nursing* and *yellow fever* with the largest differences. Compared to word embeddings, the combination performs better in most cases. Despite the combination performing better compared to LSTM, LSTM outperforms largely the combination in ambiguous words such as *borrelia*, *cement* or *WT1*.

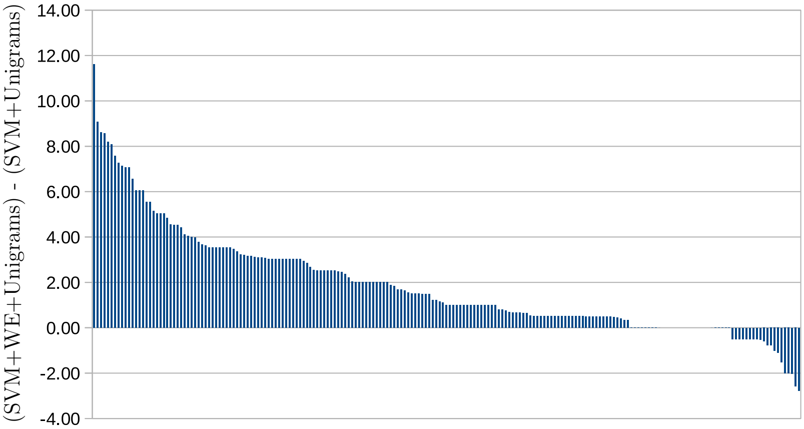

Fig. 7. MSH WSD set difference in accuracy per ambiguous word between the combination of word embeddings with unigrams (WE + Unigrams in Table 6) versus just using SVM and unigrams (Table 2) sorted in descending order.

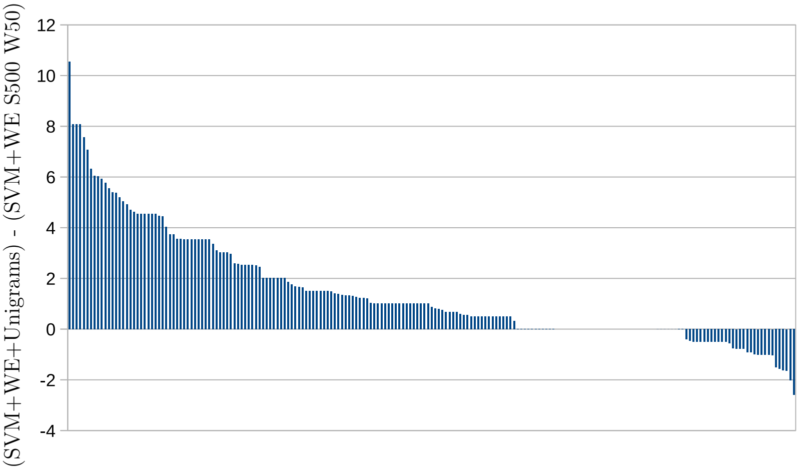

Fig. 8. MSH WSD set difference in accuracy per ambiguous word between the combination of word embeddings with unigrams (WE + Unigrams in Table 6) versus average word embeddings with vectors size 500 and window 50 and SVM (Table 4) sorted in descending order.

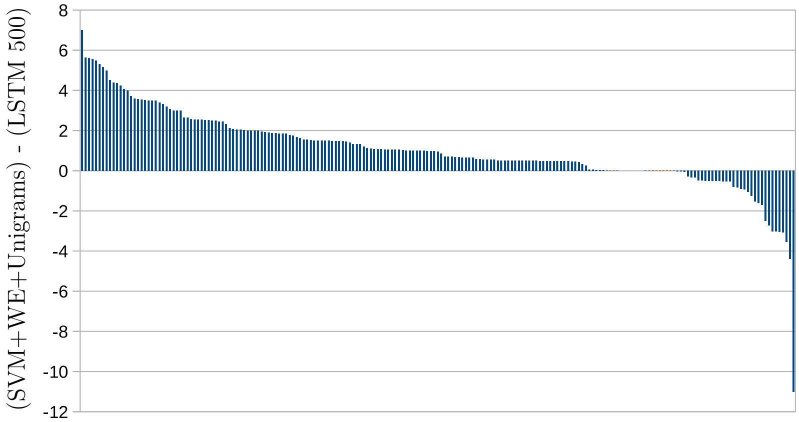

Fig. 9. MSH WSD set difference in accuracy per ambiguous word between the combination of word embeddings with unigrams (WE + Unigrams in Table 6) versus LSTM with vector size 500 and window 50 (LSTM 500 in Table 4) sorted in descending order.

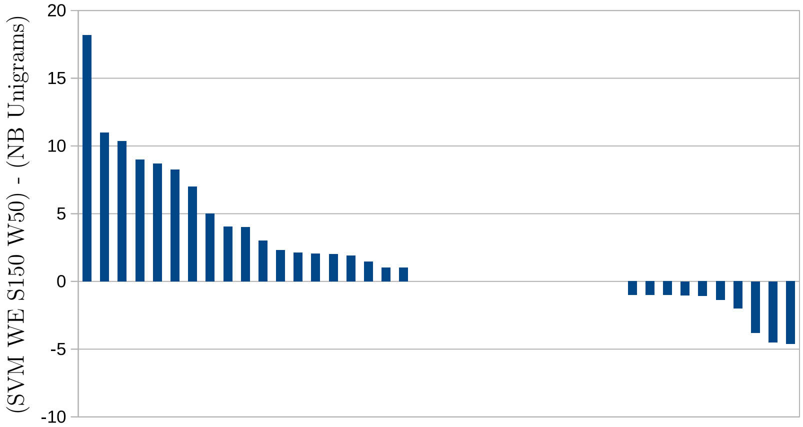

Fig. 10. NLM WSD set difference in accuracy per ambiguous word between the word embeddings with SVM (SVM WE S150 W50 in Table 5) versus unigrams and Naïve Bayes (NB Unigrams in Table 5) sorted in descending order.

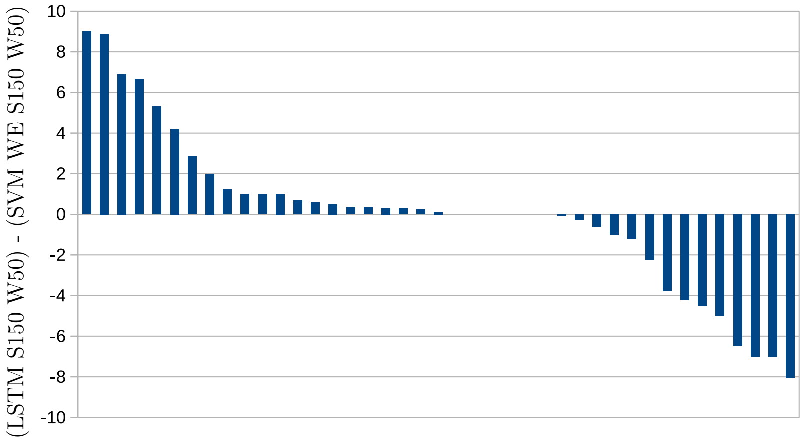

Fig. 11. NLM WSD set difference in accuracy per ambiguous word between LSTM (LSTM S150 W50 in Table 6) versus word embeddings with SVM (SVM WE S150 W50 in Table 4) sorted in descending order.

The same analysis was done on the NLM WSD set. Feature combination does not seem to improve compared to unigrams, even when combined with word embeddings. Fig. 10 shows the difference in accuracy between SVM with word embeddings and Naïve Bayes and unigrams. Ambiguous word *reduction* has over 18 points difference, this has two senses,[1] one as *Natural phenomenon or*

[1] https://wsd.nlm.nih.gov/info/wsd.cases_Final.pdf.

process and another one as *Health Care Activity*. Word embeddings might provide means to understand the context of the ambiguous word as either related to one sense or the other. Fig. 11 shows the differences in performance between LSTM WE S150 W50 and SVM with word embeddings. Differences are not as large as in the previous figure. With respect to the ambiguous word *reduction*, both methods have the same performance.

6. Conclusions and future work

The combination of unigrams and word embeddings with SVM sets a new state of the art performance with the MSH WSD data set with an accuracy of 95.97, but this is not the case for the NLM WSD set. For the NLM WSD set, LSTM with word embeddings provides the better accuracy followed by non-deep-network learning algorithms with word embeddings and feature combination does not seem to improve performance. On both sets, word embeddings and LSTM improve over single feature sets.

Using representations based on word embeddings reduce the dimensionality of the bag-of-word vectors and could be used in functions for probability estimation, which could be used in unsupervised methods based on probabilistic graphical models [17].

Recent work has studied the use of not only generation of vectors at the word level but at the document level, for instance for text categorization [24,22] and it would be interesting to see the performance of their methods on the WSD problem presented in this work.

LSTM has been trained using a reduced number of examples and could benefit from using a larger set. Training has been done on examples from the MSH WSD and NLM WSD data sets. Following the procedure used to generate the MSH WSD data set, it would be possible to extend the training set.

Supervised methods perform typically better compared to knowledge-based approaches but require training data, which limits its usability. The outcome of this work is relevant to understand how word embeddings support biomedical word sense disambiguation and encourages extending the current work in the knowledge-based scenario.

Acknowledgements

The author would like to thank Dr. Bridget McInnes for motivating this work and providing ideas and semantic annotation for the examples.

Conflict of interest

I declare that I have no conflict of interest.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.jbi.2017.08.001.

References

[1] E. Agirre, P. Edmonds, Word sense disambiguation: Algorithms and applications, vol. 33, Springer Science & Business Media, 2007.

[2] E. Agirre, A. Soroa, M. Stevenson, Graph-based word sense disambiguation of biomedical documents, Bioinformatics 26 (22) (2010) 2889–2896.

[3] A.R. Aronson, F.-M. Lang, An overview of metamap: historical perspective and recent advances, J. Am. Med. Inform. Assoc. 17 (3) (2010) 229–236.

[4] Y. Bengio, R. Ducharme, P. Vincent, C. Janvin, A neural probabilistic language model, J. Machine Learning Res. 3 (2003) 1137–1155.

[5] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166.

[6] S. Brody, M. Lapata, Bayesian word sense induction, in: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 2009, pp. 103–111.

[7] R. Chasin, A. Rumshisky, O. Uzuner, P. Szolovits, Word sense disambiguation in the clinical domain: a comparison of knowledge-rich and knowledge-poor unsupervised methods, J. Am. Med. Inform. Assoc. 21 (5) (2014) 842–849.

[8] X. Chen, Z. Liu, M. Sun, A unified model for word sense representation and disambiguation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, October 2014, pp. 1025–1035. <http://www.aclweb.org/anthology/D14-1110>.

[9] P.R. Cohen, Empirical methods for artificial intelligence, IEEE Intell. Syst. (6) (1996) 88.

[10] R. Collobert, K. Kavukcuoglu, C. Farabet, Torch7: a matlab-like environment for machine learning, in: BigLearn, NIPS Workshop. No. EPFL-CONF-192376, 2011.

[11] A.M. Dai, Q.V. Le, Semi-supervised sequence learning, in: Advances in Neural Information Processing Systems, 2015, pp. 3079–3087.

[12] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Machine Learning Res. 12 (2011) 2121–2159.

[13] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using wikipedia-based explicit semantic analysis, in: IJCAI, vol. 7, 2007, pp. 1606–1611.

[14] A. Graves, Generating sequences with recurrent neural networks, arXiv:1308.0850, 2013.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, ACM SIGKDD Explor. Newslett. 11 (1) (2009) 10–18.

[16] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[17] A. Jimeno Yepes, R. Berlanga, Knowledge based word-concept model estimation and refinement for biomedical text mining, J. Biomed. Inform. 53 (2015) 300–307.

[18] A. Jimeno Yepes, L. Plaza, J. Carrillo-de Albornoz, J.G. Mork, A.R. Aronson, Feature engineering for medline citation categorization with mesh, BMC Bioinform. 16 (1) (2015) 113.

[19] A.J. Jimeno-Yepes, B.T. McInnes, A.R. Aronson, Exploiting mesh indexing in medline to generate a data set for word sense disambiguation, BMC Bioinform. 12 (1) (2011) 223.

[20] G.H. John, P. Langley, Estimating continuous distributions in bayesian classifiers, in: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.

[21] M. Kågebäck, H. Salomonsson, Word sense disambiguation using a bidirectional lstm, arXiv:1606.03568, 2016.

[22] A. Kosmopoulos, I. Androutsopoulos, G. Paliouras, Biomedical semantic indexing using dense word vectors in bioasq, J. BioMed. Semant. Suppl. BiosMed. Inform. Retr., 2015.

[23] J.H. Lau, P. Cook, D. McCarthy, D. Newman, T. Baldwin, Word sense induction for novel sense detection, in: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 2012, pp. 591–601.

[24] Q.V. Le, T. Mikolov, Distributed representations of sentences and documents, arXiv:1405.4053, 2014.

[25] J. Li, M.-T. Luong, D. Jurafsky, A hierarchical neural autoencoder for paragraphs and documents, arXiv:1506.01057, 2015.

[26] B.T. McInnes, T. Pedersen, J. Carlis, Using umls concept unique identifiers (cuis) for word sense disambiguation in the biomedical domain, AMIA Annual Symposium Proceedings, vol. 2007, American Medical Informatics Association, 2007, p. 533.

[27] B.T. McInnes, T. Pedersen, Y. Liu, G.B. Melton, S.V. Pakhomov, Knowledge-based method for determining the meaning of ambiguous biomedical terms using information content measures of similarity, AMIA Annual Symposium Proceedings, vol. 895, American Medical Informatics Association, 2011.

[28] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv:1301.3781, 2013.

[29] R. Navigli, Word sense disambiguation: a survey, ACM Comput. Surveys (CSUR) 41 (2) (2009) 10.

[30] R. Navigli, S. Faralli, A. Soroa, O. de Lacalle, E. Agirre, Two birds with one stone: learning semantic models for text categorization and word sense disambiguation, in: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, ACM, 2011, pp. 2317–2320.

[31] T. Pedersen, The effect of different context representations on word sense discrimination in biomedical texts, in: Proceedings of the 1st ACM International Health Informatics Symposium, ACM, 2010, pp. 56–65.

[32] J. Platt, et al., Sequential minimal optimization: a fast algorithm for training support vector machines, 1998.

[33] S. Rothe, H. Schütze, Autoextend: extending word embeddings to embeddings for synsets and lexemes, 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). Association for Computational Linguistics, Beijing, China, July 2015, pp. 1793–1803. <http://www.aclweb.org/anthology/P15-1173>.

[34] L. Smith, T. Rindflesch, W.J. Wilbur, et al., Medpost: a part-of-speech tagger for biomedical text, Bioinformatics 20 (14) (2004) 2320–2321.

[35] M. Stevenson, Y. Guo, R. Gaizauskas, D. Martinez, Disambiguation of biomedical text using diverse sources of information, BMC Bioinformatics 9 (11) (2008) 1.

[36] H. Sugawara, H. Takamura, R. Sasano, M. Okumura, Context representation with word embeddings for wsd, in: International Conference of the Pacific Association for Computational Linguistics, Springer, 2015, pp. 108–119.

[37] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[38] K. Taghipour, H.T. Ng, 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, Association for Computational Linguistics, Denver, Colorado, 2015, pp. 314–323. http://www.aclweb.org/anthology/N15-1035 .

[39] M. Weeber, J.G. Mork, A.R. Aronson, Developing a test collection for biomedical word sense disambiguation, in: Proceedings of the AMIA Symposium, American Medical Informatics Association, 2001, p. 746.

[40] D. Yuan, R. Doherty, J. Richardson, C. Evans, E. Altendorf, Word sense disambiguation with neural language models, arXiv:1603.07012, 2016.

[41] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Advances in Neural Information Processing Systems, 2015, pp. 649–657.

[42] Z. Zhong, H.T. Ng, It makes sense: a wide-coverage word sense disambiguation system for free text, in: Proceedings of the ACL 2010 System Demonstrations, Association for Computational Linguistics, 2010, pp. 78–83.