Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation

Anonymous NAACL submission

Abstract

Named Entity Disambiguation (NED) refers to the task of resolving multiple named entity mentions in a document to their correct references in a knowledge base (KB) (e.g., Wikipedia). In this paper, we propose a novel embedding method specifically designed for NED. The proposed method *jointly* maps words and entities into the same continuous vector space. We extend the *skip-gram* model by using two models. The *KB graph model* learns the relatedness of entities using the link structure of the KB, whereas the *anchor context model* aims to align vectors such that similar words and entities occur close to one another in the vector space by leveraging KB anchors and their context words. By combining contexts based on the proposed embedding with standard NED features, we achieved state-of-the-art accuracy of 93.1% on the standard CoNLL dataset and 85.2% on the TAC 2010 dataset. Our code and pre-trained vectors will be made available online.

1 Introduction

Named Entity Disambiguation (NED) is the task of resolving ambiguous mentions of entities to their referent entities in a knowledge base (KB) (e.g., Wikipedia). NED has lately been extensively studied (Cucerzan, 2007; Mihalcea and Csomai, 2007; Milne and Witten, 2008b; Ratinov et al., 2011) and used as a fundamental component in numerous tasks, such as information extraction, knowledge base population (McNamee and Dang, 2009; Ji et al., 2010), and semantic search (Blanco et al., 2015). We use Wikipedia as our KB in this paper.

The main difficulty in NED is ambiguity in the meaning of entity mentions. For example, the mention "Washington" in a document can refer to various entities, such as the state, or the capital in the US, the actor Denzel Washington, the first US president George Washington, and so on. In order to resolve these ambiguous mentions into references to the correct entities, early approaches focused on modeling *textual* context, such as the similarity between contextual words and encyclopedic descriptions of a candidate entity (Bunescu and Pasca, 2006; Mihalcea and Csomai, 2007). Most state-of-the-art methods use more sophisticated *global* approaches, wherein all mentions in a document are simultaneously disambiguated based on global *coherence* among disambiguation decisions.

Word embedding methods are also becoming increasingly popular (Mikolov et al., 2013a; Mikolov et al., 2013b; Pennington et al., 2014). These involve learning continuous vector representations of words from large, unstructured text corpora. The vectors are designed to capture the semantic similarity of words when similar words are placed near one another in a relatively low-dimensional vector space.

In this paper, we propose a method to construct a novel embedding that *jointly* maps words and entities into the same continuous vector space. In this model, similar words and entities are placed close to one another in a vector space. Hence, we can measure the similarity between any pairs of items (i.e., words, entities, and a word and an entity) by simply computing their cosine similarity. This enables us to easily measure the contextual information for NED, such as the similarity between a context word and a candidate entity, and the relatedness of entities required to model coherence.

Our model is based on the skip-gram model (Mikolov et al., 2013a; Mikolov et al., 2013b), a recently proposed embedding model that learns to predict each context word given the target word. Our model consists of the following three models based on the skip-gram model: 1) the conventional skip-gram model that learns to predict the neighboring words given the target word in the text corpora, 2) the *KB graph model* that learns to estimate the neighboring entities given the target entity in the link graph of the KB, and 3) the *anchor context model* that learns to predict the neighboring words given the target entity using the anchors and their context words in the KB. By jointly optimizing these models, our method simultaneously learns the embedding of words and entities.

Based on our proposed embedding, we also develop a straightforward NED method that computes two contexts using the proposed embedding: textual context similarity, and coherence. Textual context similarity is measured based on vector similarity between an entity and words in the document. Coherence is measured based on the relatedness between the target entity and other entities in the document. Our NED method combines these contexts with several standard features (e.g., prior probability) using supervised machine learning.

We tested the proposed method using two standard NED datasets: the CoNLL dataset and the TAC 2010 dataset. Experimental results revealed that our method outperformed state-of-the-art methods on both datasets by significant margins. Moreover, we conducted experiments to separately assess the quality of the vector representations of words and entities using several standard word similarity datasets and an entity relatedness dataset, and discovered that our method successfully learns the quality representations of words and entities.

2 Joint Embedding of Words and Entities

In this section, we first describe the conventional skip-gram model for learning word embedding. We then explain our method to construct an embedding that jointly maps words and entities into the same continuous d-dimensional vector space. We extend the skip-gram model by adding the *KB graph model* and the *anchor context model*.

2.1 Skip-gram Model for Word Similarity

The training objective of the skip-gram model is to find word representations that are useful to predict context words given the target word. Formally, given a sequence of T words $w_1, w_2, ..., w_T$, the model aims to maximize the following objective function:

$$\mathcal{L}_w = \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{t+j}|w_t) \qquad (1)$$

where c is the size of the context window, w_t denotes the target word, and w_{t+j} is its context word. The conditional probability $P(w_{t+j}|w_t)$ is computed using the following softmax function:

$$P(w_{t+j}|w_t) = \frac{\exp(\mathbf{V}_{w_t}{}^{\top}\mathbf{U}_{w_{t+j}})}{\sum_{w \in W} \exp(\mathbf{V}_{w_t}{}^{\top}\mathbf{U}_w)} \qquad (2)$$

where W is a set containing all words in the vocabulary, and $\mathbf{V}_w \in \mathbb{R}^d$ and $\mathbf{U}_w \in \mathbb{R}^d$ denote the vectors of word w in matrices $\mathbf{V}$ and $\mathbf{U}$, respectively.

The skip-gram model is trained to optimize the above function $\mathcal{L}_w$, and $\mathbf{V}$ are used as the resulting vector representations of words.

2.2 Extending the Skip-gram Model

We extend the skip-gram model to learn the vector representations of entities. We expand matrices $\mathbf{V}$ and $\mathbf{U}$ to include the vectors of entities $\mathbf{V}_e \in \mathbb{R}^d$ and $\mathbf{U}_e \in \mathbb{R}^d$ in addition to the vectors for words.

2.2.1 KB Graph Model

We use an internal link structure in KB to enable the model to learn the relatedness between pairs of entities. Wikipedia Link-based Measure (WLM) (Milne and Witten, 2008a) is a method to measure entity relatedness based on its link structure. It has been used as a standard method to compute the relatedness of entities for modeling coherence in past NED studies. The relatedness between two entities is computed using the following function:

$$WLM(e_1, e_2) = 1 - \frac{\log \max(|C_{e_1}|,|C_{e_2}|) - \log |C_{e_1} \cap C_{e_2}|}{\log |E| - \log \min(|C_{e_1}|,|C_{e_2}|)} \qquad (3)$$

where E is the set of all entities in KB and C_e is the set of entities with a link to an entity e. Intuitively, WLM assumes that entities with similar incoming links are related. Despite its simplicity, WLM yields state-of-the-art performance (Hoffart et al., 2012).

Inspired by WLM, the KB graph model simply learns to place entities with similar incoming links near one another in the vector space. We formalize this as the following objective function:

$$\mathcal{L}_e = \sum_{e_i \in E} \sum_{e_o \in C_{e_i}, e_i \neq e_o} \log P(e_o|e_i) \tag{4}$$

We compute the conditional probability $P(e_o|e_i)$ using the following softmax function:

$$P(e_o|e_i) = \frac{\exp({\mathbf{V}_{e_i}}^\top \mathbf{U}_{e_o})}{\sum_{e \in E} \exp({\mathbf{V}_{e_i}}^\top \mathbf{U}_e)} \tag{5}$$

We train the model to predict the incoming links C_e given an entity e. Therefore, C_e plays a similar role to context words in the skip-gram model.

2.2.2 Anchor Context Model

If we add only the KB graph model to the skip-gram model, the vectors of words and entities do not interact, and can be placed in different subspaces of the vector space. To address this issue, we introduce the anchor context model to place similar words and entities near one another in the vector space.

The idea underlying this model is to leverage KB anchors and their context words to train the model. As mentioned in Section 1, we use Wikipedia as a KB. It contains many internal anchors that can be safely treated as unambiguous occurrences of referent KB entities. By using these anchors, we can easily obtain many occurrences of entities and their corresponding context words directly from the KB.

As in the skip-gram model, we simply train the model to predict the context words of an entity pointed to by the target anchor. The objective function is as follows:

$$\mathcal{L}_a = \sum_{(e_i, Q) \in A} \sum_{w_o \in Q} \log P(w_o|e_i) \tag{6}$$

where A denotes a set of anchors in the KB, each of which contains a pair of a referent entity e_i and a set of its context words Q. Here, Q contains the previous c words and the next c words. Note that $|A|$ equals the number of internal anchors in the KB. As in past models, the conditional probability $P(w_o|e_i)$ is computed using the softmax function:

$$P(w_o|e_i) = \frac{\exp({\mathbf{V}_{e_i}}^\top \mathbf{U}_{w_o})}{\sum_{w \in W} \exp({\mathbf{V}_{e_i}}^\top \mathbf{U}_w)} \tag{7}$$

Using the proposed model, we align the vector representations of words and entities by placing words and entities with similar context words close to one another in the vector space.

2.3 Training

Considering the three model components mentioned above, we propose the following objective function by linearly combining the above objective functions:

$$\mathcal{L} = \mathcal{L}_w + \mathcal{L}_e + \mathcal{L}_a \tag{8}$$

The training of the model is intended to maximize the above function, and the resulting matrix $\mathbf{V}$ is used to embed words and entities.

One of the problems in training our model is that the normalizers contained in the softmax functions $P(w_{t+j}|w_t)$, $P(e_o|e_i)$, and $P(w_o|e_i)$ are computationally very expensive because they involve summation over all words W or entities E. To address this problem, we use *negative sampling (NEG)* (Mikolov et al., 2013b) to convert original objective functions into computationally feasible ones. NEG is defined by the following objective function:

$$\log \sigma({\mathbf{V}_{w_t}}^\top \mathbf{U}_{w_{t+j}}) + \sum_{i=1}^{g} \mathbb{E}_{w_i \sim P_{neg}(w)} \Big[\log \sigma(-{\mathbf{V}_{w_t}}^\top \mathbf{U}_{w_i}) \Big] \tag{9}$$

where $\sigma(x) = 1/(1+\exp(-x))$ and g is the number of negative samples. We replace the $\log P(w_{t+j}|w_t)$ term in Eq. (1) with the above objective function. Consequently, the objective function is transformed from that in Eq. (1) to a simple objective function of the binary classification to distinguish the observed word w_t from words drawn from noise distribution $P_{neg}(w)$. We also replace $\log P(e_o|e_i)$ in Eq. (4) and $\log P(w_o|e_i)$ in Eq. (6) in the same manner.

Note that NEG takes a negative distribution $P_{neg}(w)$ as a free parameter. Following (Mikolov et al., 2013b), we use the unigram distribution of words ($U(w)$) raised to the $3/4^{th}$ power (i.e., $U(w)^{3/4}/Z$, where Z is a normalization constant) in the skip-gram model and the anchor context model. In the KB graph model, we use a uniform distribution over KB entities E as the negative distribution.

We use Wikipedia to train all the above models. Optimization is carried out simultaneously to maximize the transformed objective function by iterating over Wikipedia pages several times. We use stochastic gradient descent (SGD) for the optimization. The optimization is performed using a multiprocess-based implementation of our model using Python, Cython, and NumPy configured with OpenBLAS with storing matrices $\mathbf{V}$ and $\mathbf{U}$ in the shared memory. To improve speed, we decide not to introduce locks to the shared matrices.

3 Named Entity Disambiguation Using Embedding

In this section, we explain our NED method using our proposed embedding. Let us formally define the task. Given a set of entity mentions $M = \{m_1, m_2, ..., m_N\}$ in a document d with an entity set $E = \{e_1, e_2, ..., e_K\}$ in the KB, the task is defined as resolving mentions (e.g., "Washington") into their referent entities (e.g., Washington D.C.).

We introduce two measures that have been frequently observed in past NED studies: *entity prior* $P(e)$ and *prior probability* $P(e|m)$. We define entity prior $P(e) = |A_{e,*}|/|A_{*,*}|$ where $A_{*,*}$ denotes all anchors in the KB and $A_{e,*}$ is the set of anchors that point to entity e. Prior probability is defined as $P(e|m) = |A_{e,m}|/|A_{*,m}|$ where $A_{*,m}$ represents all anchors with the same surface as mention m in KB and $A_{e,m}$ is a subset of $A_{*,m}$ that points to entity e.

We separate the NED task into two sub-tasks: *candidate generation* and *mention disambiguation*. In candidate generation, candidates of referent entities are generated for each mention. Details of candidate generation are provided in Section 4.4.1.

3.1 Mention Disambiguation

Given a document d and mention m with its candidate referent entities $\{e_1, e_2, ..., e_k\}$ generated in the candidate generation step, the task is to disambiguate mention m by selecting the most relevant entity from the candidate entities.

The key to improving the performance of this task is to effectively model the context. We propose two novel methods to model the context using the proposed embedding. Further, we combine these two models with several standard NED features using supervised machine learning.

3.1.1 Modeling Textual Context

Textual context is designed based on the assumption that an entity is more likely to appear if the context of a given mention is similar to that of the entity.

We propose a method to measure the similarity between textual context and entity using the proposed embedding by first deriving the vector representation of the context and then computing the similarity between the context and the entity using cosine similarity. To derive the vector of context, we average the vectors of context words:

$$\vec{v_{c_w}} = \frac{1}{|W_{c_m}|} \sum_{w \in W_{c_m}} \vec{v_w} \qquad (10)$$

where W_{c_m} is a set of the context words of mention m and $\vec{v_w} \in \mathbf{V}$ denotes the vector representation of word w. We use all noun words in document d as context words.[1] Moreover, we ignore a context word if the surface of mention m contains it.

We then measure the similarity between candidate entity and the derived textual context by using cosine similarity between $\vec{v_{c_w}}$ and the vector of entity $\vec{v_e}$.

3.1.2 Modeling Coherence

It has been revealed that effectively modeling coherence in the assignment of entities to mentions is important for NED. However, this is a chicken-and-egg problem because the assignment of entities to mentions, which is required to measure coherence, is not possible prior to performing NED.

To address this problem, we introduce a simple *two-step* approach: we first train the machine learning model using the coherence score among unambiguous mentions[2], in addition to other features, and then retrain the model using the coherence score among the predicted entity assignments instead.

To estimate coherence, we first calculate the vector representation of the context entities and measure the similarity between the vector of the context entities and that of the target entity e. Note that context entities are unambiguous entities in the first step, and predicted entities are used instead in the second step.

To derive the vector representation of context entities, we average their vector representations:

$$\vec{v_{c_e}} = \frac{1}{|E_{c_m}|} \sum_{e^* \in E_{c_m}} \vec{v_{e^*}} \qquad (11)$$

[1] We used Apache OpenNLP tagger to detect nouns. https://opennlp.apache.org/

[2] We consider that mention m unambiguously refers to entity e if its prior probability $P(e|m)$ is greater than 0.95.

where E_{c_m} denotes the set of context entities described above.

To estimate the coherence score, we again use cosine similarity between the vector of entity $\vec{v}_e$ and that of context entities $\vec{v}_{c_e}$.

3.1.3 Learning to Rank

To combine the proposed contextual information described above with standard NED features, we employ a method of supervised machine learning to rank the candidate entities given mention m and document d.

In particular, we use Gradient Boosted Regression Trees (GBRT) (Friedman, 2001), a state-of-the-art point-wise learning-to-rank algorithm widely used for various tasks, which has been recently adopted for the sort of tasks for which we employ it here (Meij et al., 2012). GBRT consists of an ensemble of regression trees, and predicts a relevance score given an instance. We use the GBRT implementation in *scikit-learn*[3] and the logistic loss is used as the loss function. The main parameters of GBRT are the number of iterations η, the learning rate β, and the maximum depth of the decision trees ξ.

With regard to the features of machine learning, we first use prior probability ($P(e|m)$) and entity prior ($P(e)$). Further, we include a feature representing the maximum prior probability of the candidate entity e of all mentions in the document. We also add the number of entity candidates for mention m as a feature. The above set of four features is called *base* features in the rest of the paper.

We also use several *string similarity* features used in past work on NED (Meij et al., 2012). These features aim to capture the similarity between the title of entity e and the surface of mention m, and consist of the edit distance, whether the title of entity e exactly equals or contains the surface of mention m, and whether the title of entity e starts or ends with the surface of mention m.

Finally, we include contextual features measured using the proposed embedding. We use cosine similarity between the candidate entity and the textual context (see Section 3.1.1), and similarity between an entity and contextual entities (see Section 3.1.2). Furthermore, we include the rank of entity e among candidate entities of mention m, sorted according to these two similarity scores in descending order.

[3]http://scikit-learn.org/

	WordSim-353	MC	RG
Our Method	0.66	**0.78**	**0.77**
Skip-gram	**0.67**	0.77	0.76

Table 1: Results of the word similarity task.

4 Experiments

In this section, we describe the setup and results of our experiments. In addition to experiments on the NED task, we conducted two experiments—one involving a *word similarity* and another involving an *entity relatedness*—in order to test the effectiveness of our method in capturing pairwise similarity between pairs of words as well as pairs of entities. We first describe the details of the training of the embedding and then present the experimental results.

4.1 Training for the Proposed Embedding

To train the proposed embedding, we used the December 2014 version of the Wikipedia dump[4]. We first removed the pages for navigation, maintenance, and discussion, and used the remaining 4.9 million pages. We parsed the Wikipedia pages and extracted text and anchors from each page. We further tokenized the text using the *Apache OpenNLP* tokenizer. We also filtered out rare words that appeared fewer than five times in the corpus. We thus obtained approximately 2 billion tokens and 73 million anchors. The total number of words and entities in the embedding were approximately 2.1 million and 5 million, respectively. Consequently, the number of rows of matrices $\mathbf{V}$ and $\mathbf{U}$ were 7.1 million.

The number of dimensions d of the embedding was set to 500. Following (Mikolov et al., 2013b), we also used learning rate $\alpha = 0.025$ which linearly decreased with the iterations of the Wikipedia dump. Regarding the other parameters, we set the size of the context window $c = 10$ and the negative samples $g = 30$. The model was trained online by iterating over pages in the Wikipedia dump 10 times. The training lasted approximately five days using a server with a 40-core CPU on Amazon EC2.

[4]The dump was retrieved from Wikimedia Downloads. http://dumps.wikimedia.org/

	NDCG@1	NDCG@5	NDCG@10	MAP
Our Method	**0.59**	**0.56**	**0.59**	**0.52**
WLM	0.54	0.52	0.55	0.48

Table 2: Results of the entity relatedness task.

4.2 Word Similarity

In order to test the quality of vector representations of words, we used three standard word similarity datasets: the *WordSim-353* dataset (Finkelstein et al., 2002), the *MC* dataset (Miller and Charles, 1991), and the *RG* dataset (Rubenstein and Goodenough, 1965) that contain 353, 65, and 30 word pairs, respectively. Each word pair has a *gold-standard* similarity score assigned by human judges.

We used cosine similarity to calculate similarity score between any pair of words. Following past work, we computed the correlation between similarity scores through human judgments on a set of word pairs using Spearman's rank correlation coefficient. Here, we adopted the skip-gram model as baseline.

We used our implementation to train the skip-gram model. Furthermore, the following parameters were used to train the model: $d = 500$, $c = 10$, $g = 30$, and $\alpha = 0.025$. We trained the model by iterating over the Wikipedia dump 10 times.

Table 1 shows the results. Compared to the skip-gram model, our method performed comparably on the WordSim-353 dataset, and slightly better on other datasets, thus showing that the proposed extension to the skip-gram model can be also beneficial for improving word representations.

4.3 Entity Relatedness

To test the quality of the vector representation of entities, we conducted an experiment using a dataset for entity relatedness created by Ceccarelli et al. (Ceccarelli et al., 2013). The dataset consists of training, test, and validation sets, and we only use the test set. The test set contains 3,314 entities, where each entity has 91 candidate entities with *gold-standard* labels indicating whether the two entities are related. Following (Huang et al., 2015), we obtained the ranked order of the candidate entities using cosine similarity between the target entity and each of the candidate entities, and computed the two standard measures: normalized discounted cumulative gain (NDCG) (Järvelin and Kekäläinen, 2002) and mean average precision (MAP) (Manning et al., 2008). We adopted WLM as baseline.

Table 2 shows the results. The score for WLM was obtained from Huang et al. (Huang et al., 2015). Our method clearly outperformed WLM. The results show that our method accurately captures pairwise entity relatedness.

4.4 Named Entity Disambiguation

4.4.1 Setup

We now explain our experimental setup for the NED task. We tested the performance of our proposed method on two standard NED datasets: the *CoNLL* dataset and the *TAC 2010* dataset. The details of these datasets are provided below. Moreover, as with the corpus used in the embedding, we used the December 2014 version of the Wikipedia dump as the referent KB, and to derive the prior probability as well as the entity prior.

To find the best parameters for our machine learning model, we ran a parameter search on the CoNLL development set. We used $\eta = 10,000$ trees, and tested all combinations of the learning rate $\beta = \{0.01, 0.02, 0.03, 0.05\}$ and the maximum depth of the decision trees $\xi = \{3, 4, 5\}$. We computed their accuracy on the dataset, and found that the parameters did not significantly affect performance (1.0% at most). We used $\beta = 0.02$ and $\xi = 4$ which yielded the best performance.

CoNLL The CoNLL dataset is a popular NED dataset constructed by Hoffart et al. (Hoffart et al., 2011). The dataset is based on NER data from the CoNLL 2003 shared task, and consists of training, development, and test sets, containing 946, 216, and 231 documents, respectively. We trained our machine learning model using the training set and reported its performance using the test set. We also used the development set for the parameter tuning described above. Following (Hoffart et al., 2011), we only used 27,816 mentions with valid entries in the KB and reported the standard micro- (aggregates over all mentions) and macro- (aggregates over all documents) accuracies of the top-ranked candidate entities to assess disambiguation performance. For candidate generation, we used a public dataset[5] built by Pershina et al. (Pershina et al., 2015).

[5]`https://github.com/masha-p/PPRforNED`

TAC 2010 The TAC 2010 dataset is another popular NED dataset constructed for the Text Analysis Conference (TAC)[6] (Ji et al., 2010). The dataset is based on news articles from various agencies and Web log data, and consists of a training and a test set containing 1,043 and 1,013 documents, respectively. Following past work (He et al., 2013; Chisholm and Hachey, 2015), we used mentions only with a valid entry in the KB, and reported the micro-accuracy score of the top-ranked candidate entities. We trained our model using the training set and assessed its performance using the test set. We trained our model using the training set and assessed its performance using the test set. Consequently, we evaluated our model on 1,020 mentions contained in the test set. For candidate generation, we used a dictionary that was directly built from the Wikipedia dump mentioned previously. Similar to past work, we retrieved possible mention surfaces of an entity from (1) the title of the entity, (2) the title of another entity redirecting to the entity, and (3) the names of anchors that point to the entity. Further, we retained the top 50 candidates through their entity priors for computational efficiency.

4.4.2 Comparison with State-of-the-art Methods

We compared our method with the following recently proposed state-of-the-art methods:

- Hoffart et al. (Hoffart et al., 2011) is a graph-based approach that finds a dense subgraph of entities in a document to address NED.
- He et al. (He et al., 2013) uses deep neural networks to derive the representations of entities and mention contexts and applies them to NED.
- Chisholm and Hachey (Chisholm and Hachey, 2015) uses a Wikilinks dataset (Singh et al., 2012) to improve the performance of NED.

4.4.3 Results

Table 3 shows the experimental results of our proposed method as well as those of state-of-the-art methods. Our proposed method achieved a 93.1% micro-accuracy and 92.6% macro-accuracy on the CoNLL dataset, and 85.2% micro-accuracy on the TAC 2010 dataset. Our method significantly outperformed all the other state-of-the-art methods on both datasets by significant margins.

	CoNLL (Micro)	CoNLL (Macro)	TAC10
Our Method	**93.1**	**92.6**	**85.2**
Hoffart et al., 2011	82.5	81.7	-
He et al., 2013	85.6	84.0	81.0
Chisholm & Hachey, 2015	88.7	-	80.7

Table 3: Experimental results of NED using the proposed method and state-of-the-art methods.

	Micro accuracy	Macro accuracy
CoNLL:		
Base	85.4	87.4
+String similarity	85.8	87.8
+Textual context	90.9	92.4
+Coherence	91.4	92.1
Two-step	**93.1**	**92.6**
TAC 2010:		
Base	80.1	-
+String similarity	81.7	-
+Textual context	84.6	-
+Coherence	**85.5**	-
Two-step	85.2	-

Table 4: The results of our feature study.

4.4.4 Feature Study

We conducted a feature study on our method. We began with base features, added various features to our system incrementally, and reported their impact on performance. We then introduced our two-step approach to achieve the final results.

Table 4 shows the results. Surprisingly, we attained results comparable with those of most state-of-the-art methods on the both datasets by only using base features. Adding string similarity features slightly further improved performance.

We observed significant improvement when adding textual context features based on our proposed embedding. Our method outperformed other state-of-the-art methods without using coherence.

Further, coherence based on unambiguous entity mentions and our two-step approach significantly improved performance on the CoNLL dataset. However, it did not contribute to performance on the TAC

[6] http://www.nist.gov/tac/

2010 dataset. This was because of the significant difference in the density of entity mentions between the datasets. The CoNLL dataset contains approximately 20 entity mentions per document, but the TAC 2010 only contains approximately one mention per document which is unarguably insufficient to model coherence.

4.4.5 Error Analysis

We also conducted an error analysis on the CoNLL test set. We observed that approximately 48.6% errors were caused by *metonymy* mentions (Ling et al., 2015) (i.e., mentions with more than one plausible annotation). In particular, our NED method often erred when an incorrect entity was highly popular and exactly matched the mention surface (e.g., "South Africa" referring to the entity South Africa national rugby union team rather than the entity South Africa). This makes sense because our machine learning model uses the popularity statistics of the KB (i.e., prior probability and entity prior), and the string similarity between the title of the entity and the mention surface. This problem is discussed further in (Ling et al., 2015).

5 Related Work

Early NED methods addressed the problem as a well-studied *word sense disambiguation* problem (Mihalcea and Csomai, 2007). These methods primarily focused on modeling the similarity of *textual* (*local*) context. Most recent state-of-the-art methods focus on modeling *coherence* among disambiguated entities in the same document (Cucerzan, 2007; Milne and Witten, 2008b; Hoffart et al., 2011; Ratinov et al., 2011). These approaches have also been called *collective* or *global* approaches in the literature.

Learning the representations of entities for NED has been addressed in past literature. Guo and Barbosa (Guo and Barbosa, 2014) used random walks on KB graphs to construct vector representations of entities and documents to address NED. Blanco et al. (Blanco et al., 2015) proposed a method to map entities into the word embedding (i.e., Word2vec (Mikolov et al., 2013b)) space using entity descriptions in the KB and applied it for NED. He et al. (He et al., 2013) used deep neural networks to compute representations of entities and contexts of mentions directly from the KB. Similarly, Sun et al. (Sun et al., 2015) proposed a method based on deep neural networks to model representations of mentions, contexts of mentions, and entities. Huang et al. (Huang et al., 2015) also leveraged deep neural networks to learn entity representations such that the consequent pairwise entity relatedness was more suitable than of a standard method (i.e., WLM) for NED. Further, Hu et al. (Hu et al., 2015) used hierarchical information in the KB to build entity embedding and applied it to model coherence. Unlike these methods, our proposed approach involves jointly learning vector representations of entities as well as words, hence enabling the accurate computation of the semantic similarity among its items to model both the textual context and coherence.

Furthermore, in the context of *knowledge graph embedding*, another tenor of recent works has been published (Bordes et al., 2011; Socher et al., 2013; Lin et al., 2015). These methods focus on learning vector representations of entities to primarily address the *link prediction* task that aims to predict a new fact based on existing facts in KB. Particularly, Wang et al. (Wang et al., 2014) have recently revealed that the joint modeling of the embedding of words and entities can improve performance in several tasks including the link prediction task, which is somewhat analogous to our experimental results.

6 Conclusions

In this paper, we proposed an embedding method to jointly map words and entities into the same continuous vector space. Our method enables us to effectively model both *textual* and *global* contexts. Further, armed with these context models, our NED method significantly outperforms state-of-the-art NED methods.

In future work, we intend to improve our model by leveraging relevant knowledge, such as relations in a knowledge graph (e.g., Freebase). We would also like to seek applications of our proposed embedding other than NED.

The code of the proposed embedding method and the pre-trained vectors used in our experiments will be made publicly available before the conference.

References

Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and Space-Efficient Entity Linking for Queries. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 179–188.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI)*, pages 301–306.

Razvan Bunescu and Marius Pasca. 2006. Using Encyclopedic Knowledge for Named Entity Disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 9–16.

Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. 2013. Learning Relatedness Measures for Entity Linking. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM)*, pages 139–148.

Andrew Chisholm and Ben Hachey. 2015. Entity Disambiguation with Web Links. *Transactions of the Association for Computational Linguistics*, 3:145–156.

Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing Search in Context: The Concept Revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Jerome H. Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232.

Zhaochen Guo and Denilson Barbosa. 2014. Entity Linking with a Unified Semantic Representation. In *Proceedings of the 23rd International Conference on World Wide Web (WWW)*, pages 1305–1310.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning Entity Representation for Entity Disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 2: Short Papers)*, pages 30–34.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 782–792.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: Keyphrase Overlap Relatedness for Entity Disambiguation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*, pages 545–554.

Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric Xing. 2015. Entity Hierarchy Embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP) (Volume 1: Long Papers)*, pages 1292–1300.

Hongzhao Huang, Larry Heck, and Heng Ji. 2015. Leveraging Deep Neural Networks and Knowledge Graphs for Entity Disambiguation. *CoRR*, abs/1504.0.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the TAC 2010 Knowledge Base Population Track. In *Proceeding of Text Analytics Conference (TAC)*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design Challenges for Entity Linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

P McNamee and HT Dang. 2009. Overview of the TAC 2009 Knowledge Base Population Track. In *Proceeding of Text Analysis Conference (TAC)*.

Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. 2012. Adding Semantics to Microblog Posts. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 563–572.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (CIKM)*, pages 233–242.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–12.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

George A. Miller and Walter G. Charles. 1991. Contextual Correlates of Semantic Similarity. *Language and Cognitive Processes*, 6(1):1–28.

David Milne and Ian H. Witten. 2008a. An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links. In *Proceedings of the First AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI)*.

David Milne and Ian H. Witten. 2008b. Learning to Link with Wikipedia. In *Proceeding of the 17th ACM Conference on Information and Knowledge Management (CIKM)*, pages 509–518.

Jeffrey Pennington, Richard Socher, and Christopher D 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.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized Page Rank for Named Entity Disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 238–243.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 1375–1384.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual Correlates of Synonymy. *Communications of the ACM*, 8(10):627–633.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A Large-scale Cross-Document Coreference Corpus Labeled via Links to Wikipedia. Technical Report UM-CS-2012-015.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Advances in Neural Information Processing Systems (NIPS)*, pages 926–934.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1333–1339.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph and Text Jointly Embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601.