Entity Disambiguation by Knowledge and Text Jointly Embedding

Wei Fang[1,2], **Jianwen Zhang**[3], **Dilin Wang**[4], **Zheng Chen**[3], **Ming Li**[1,2]
[1]SYSU-CMU Joint Institute of Engineering,
School of Electronics and Information Technology, Sun Yat-Sen University
[2]SYSU-CMU Shunde International Joint Research Institute
{fangwei7@mail2, liming46@mail}.sysu.edu.cn
[3]Microsoft, Redmond, WA
{jiazhan, zhengc}@microsoft.com
[4]Computer Science, Dartmouth College
dilin.wang.gr@dartmouth.edu

Abstract

For most entity disambiguation systems, the secret recipes are feature representations for mentions and entities, most of which are based on Bag-of-Words (BoW) representations. Commonly, BoW has several drawbacks: (1) It ignores the intrinsic meaning of words/entities; (2) It often results in high-dimension vector spaces and expensive computation; (3) For different applications, methods of designing handcrafted representations may be quite different, lacking of a general guideline. In this paper, we propose a different approach named EDKate. We first learn low-dimensional continuous vector representations for entities and words by jointly embedding knowledge base and text in the same vector space. Then we utilize these embeddings to design simple but effective features and build a two-layer disambiguation model. Extensive experiments on real-world data sets show that (1) The embedding-based features are very effective. Even a single one embedding-based feature can beat the combination of several BoW-based features. (2) The superiority is even more promising in a difficult set where the mention-entity prior cannot work well. (3) The proposed embedding method is much better than trivial implementations of some off-the-shelf embedding algorithms. (4) We compared our EDKate with existing methods/systems and the results are also positive.

1 Introduction

Entity disambiguation is the task of linking entity mentions in unstructured text to the corresponding entities in a knowledge base. For example, in the sentence "*Michael Jordan is newly elected as AAAI fellow*", the mention "*Michael Jordan*" should be linked to "*Michael I. Jordan*" (Berkeley Professor) rather than "*Michael Jordan*" (NBA Player). Formally, given a set of mentions $M = \{m_1, m_2, ..., m_k\}$ (specified or detected automatically) in a document d, for each mention $m_i \in M$, the task is to find the correct entity e_i in the knowledge base (KB) $\mathcal{K}$ to which the mention m_i refers.

There are various methods proposed for the problem in the past decades. But generally speaking, an entity disambiguation method is commonly composed of three stages/components. (1) Constructing representations for mentions/entities from raw data, often as the form of sparse vectors. (2) Extracting features for disambiguation models based on the representations of mentions and entities constructed in stage (1). (3) Optimizing the disambiguation model by empirically setting or learning weights on the extracted features, e.g., by training a classifier/ranker. There exist few features directly defined by heuristics, skipping the first stage. For example, string similarity or edit distance between a mention surface and an entity's canonical form (Cucerzan, 2011; Cassidy et al., 2011), and the prior probability of a mention surface being some entity, etc. However, they are the minority as it is difficult for human to design such features.

Almost all the existing methods focus on the second or the third stages while the importance of the first stage is often overlooked. The common practice to deal with the first stage of representa-

Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), pages 260–269,
Berlin, Germany, August 7-12, 2016.

tions is defining handcrafted BoW representations. For example, an entity is often represented by a sparse vector of weights on the n-grams contained in the description text of the entity, i.e., the standard Bag-of-Words (BoW) representation. TF-IDF is often used to set the weights. There are several variants for this way, e.g., using selected key phrases or Wikipedia in-links/out-links instead of all n-grams as the dimensions of the vectors (Ratinov et al., 2011). The problem is more challenging when representing a mention. The common choice is using the n-gram vector of the surrounding text. Obviously the information of the local text window is too limited to well represent a mention. In practice, there is another constraint, the representations of entities and mentions should be in the same space, i.e., the dimensions of the vectors should be shared. This constraint makes the representation design more difficult. How to define such representations and the features based on them almost become the secrete sauce of a disambiguation system. For example, Cucerzan (2007) uses Wikipedia anchor surfaces and "Category" values as dimensions and designed complex mechanisms to represent words, mentions and entities as sparse vectors on those dimensions.

BoW representations have several intrinsic drawbacks: First, the semantic meaning of a dimension is largely ignored. For example, "cat", "cats" and "tree" are equally distant under one-hot BoW representations. Second, BoW representations often introduce high dimension vector spaces and lead to expensive computation. Third, for different applications, methods of designing handcrafted representations may be quite different, lacking of a general guideline. The intuitive questions like "why using n-grams, Wikipedia links or category values as dimensions" and "why using TF-IDF as weights" are hinting us it is very likely these handcrafted representations are not the best and there should be some better representations.

In this paper we focus on the first stage, the problem of representations. Inspired by the recent works on word embedding (Bengio et al., 2003; Collobert et al., 2011; Mikolov et al., 2013a; Mikolov et al., 2013b), knowledge embedding (Bordes et al., 2011; Bordes et al., 2013; Socher et al., 2013; Wang et al., 2014b) and joint embedding KBs and texts (Wang et al., 2014a; Zhong and Zhang, 2015), we propose to learn representations for entity disambiguation. Specifically, from KBs and texts, we jointly embed entities and words into the same low-dimensional continuous vector space. The embeddings are obtained by optimizing a global objective considering all the information in the KBs and texts thus the intrinsic semantics of words and entities are believed to be preserved during the embedding. Then we design simple but effective features based on embeddings and build a two-layer disambiguation model. We conduct extensive experiments on real-word data sets and exhibit the effectiveness of our words and entities' representation.

2 Related Work

Entity Disambiguation Entity disambiguation methods roughly fall into two categories: local approaches and collective approaches. Local approaches disambiguate each mention in a document separately. For example, Bunescu and Pasca (2006) compare the context of each mention with the Wikipedia categories of an entity candidate; Milne and Witten (2008) come up with the concept "unambiguous link" and make it convenient to compute entity relatedness. Differently, collective approaches require all entities in a document "coherent" in semantic, measured by some objective functions. Cucerzan (2007) proposes a topic representation for document by aggregating topic vectors of all entity candidates in the document. Kulkarni et al. (2009) model pair-wise coherence of entity candidates for two different mentions and use hill-climbing algorithm to get a proximate solution. Hoffart et al. (2011) treat entity disambiguation as the task of finding a dense subgraph which contain all mention nodes and exactly one mention-entity edge for each mention from a large graph.

Most methods above design various representations for mentions and entities. For example, based on Wikipedia, Cucerzan (2007) uses anchor surfaces to represent entities in "context space" and use items in the category boxes to represent entities in "topic space". For mentions, he takes context words among a fixed-size window around the mention as the context vector. Kulkarni et al. (2009) exploit sets of words, sets of word counts and sets of TF-IDFs to represent entities. Ratinov et al. (2011) express entities with extensive in-links and out-links in Wikipedia.

In recent years, some works are considering

how to apply neural network to disambiguate entities from context. For example, He et al. (2013) use feed-forward network to represent context based on BoW input while Sun et al. (2015) turn to convolution network directly based on the original word2vec (Mikolov et al., 2013a). However, they pay little attention to design effective word and entity representations. In this paper, we focus on learning representative word and entity vectors for disambiguation.

Embedding Word embedding aims to learn continuous vector representation for words. Word embeddings are usually learned from unlabeled text corpus by predicting context words surrounded or predicting the current word given context words (Bengio et al., 2003; Collobert et al., 2011; Mikolov et al., 2013a; Mikolov et al., 2013b). These embeddings can usually catch syntactic and semantic relations between words.

Recently knowledge embedding also becomes popular. The goal is to embed entities and relations of knowledge graphs into a low-dimension continuous vector space while certain properties in the graph are preserved (Bordes et al., 2011; Bordes et al., 2013; Socher et al., 2013; Wang et al., 2014a; Wang et al., 2014b). To connect word embedding and knowledge embedding, (Wang et al., 2014a) propose to align these two spaces by Wikipedia anchors and names of entities. (Zhong and Zhang, 2015) conduct alignment by entities' description.

3 Disambiguation by Embedding

In this part, we first refine current joint embedding techniques to train word and entity embeddings from Freebase and Wikipedia texts for disambiguation tasks. Then in section 3.2, we design simple features based on embeddings. Finally in section 3.3, we propose a two-layer disambiguation model to balance mention-entity prior and other features.

3.1 Embeddings Jointly Learning

We mainly base the joint learning framework on (Wang et al., 2014a)'s joint model and also utilize the alignment technique from (Zhong and Zhang, 2015) to better align word and entity embeddings into a same space. Furthermore, we optimize the embedding for disambiguation from two aspects. First, we add url-anchor (entity-entity) co-occurrence from Wikipedia. Second, we refine the traditional negative sampling part to have entities in candidate list more probable to be sampled, which aims to discriminate entity candidates from each other.

3.1.1 Knowledge Model

A knowledge base $\mathcal{K}$ is usually composed of a set of triplets (h, r, t), where $h, t \in \mathcal{E}$ (the set of entities) and $r \in \mathcal{R}$ (the set of relations). Here, we follow (Wang et al., 2014a) to use $\mathbf{h}, \mathbf{r}, \mathbf{t}$ to denote the embeddings of h, r, t respectively. And score a triplet in this way:

$$z(h, r, t) = b - \frac{1}{2} \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|^2 \quad (1)$$

where b is a constant for numerical stability in the approximate optimization stage described in 3.1.5. Then normalize z and define:

$$\Pr(h|r, t) = \frac{\exp\{z(h, r, t)\}}{\sum_{\tilde{h} \in \mathcal{E}} \exp\{z(\tilde{h}, r, t)\}} \quad (2)$$

$\Pr(r|h, t)$ and $\Pr(t|h, r)$ are also defined in a similar way. And the likelihood of observing a triplet is:

$$\mathcal{L}_{triplet}(h, r, t) = \log \Pr(h|r, t) + \log \Pr(r|h, t) + \log \Pr(t|h, r) \quad (3)$$

Then the goal is to maximize the likelihood of all triplets in the whole knowledge graph:

$$\mathcal{L}_K = \sum_{(h,r,t) \in \mathcal{K}} \mathcal{L}_{triplet}(h, r, t) \quad (4)$$

3.1.2 Text Model

In text model, to be compatible with the knowledge model, a pair of co-occurrence words is scored in this way:

$$z(w, v) = b - \frac{1}{2} \|\mathbf{w} - \mathbf{v}\|^2 \quad (5)$$

where w and v represent co-occurrence of two words in a context window; $\mathbf{w}$ and $\mathbf{v}$ represent the corresponding embeddings for w and v. Then normalize $z(w, v)$ and give a probability representation:

$$\Pr(w|v) = \frac{\exp\{z(w, v)\}}{\sum_{\tilde{w} \in \mathcal{V}} \exp\{z(\tilde{w}, v)\}} \quad (6)$$

where $\mathcal{V}$ is our vocabulary. Then the goal of the text model is to maximize the likelihood of all word co-occurrence pairs:

$$\mathcal{L}_T = \sum_{(w,v)} [\log \Pr(w|v) + \log \Pr(v|w)] \quad (7)$$

3.1.3 Alignment Model

Alignment model guarantees the vectors of entities and words are in the same space, i.e., the similarity/distance between an entity vector and an word vector is meaningful. We combine all the three alignment models proposed in (Wang et al., 2014a) and (Zhong and Zhang, 2015).

Alignment by Wikipedia Anchors (Wang et al., 2014a). Mentions are replaced with the entities they link to and word-word co-occurrence becomes word-entity co-occurrence.

$$\mathcal{L}_{AA} = \sum_{(w,a),a\in\mathcal{A}} [\log \Pr(w|e_a) + \log \Pr(e_a|w)] \tag{8}$$

where $\mathcal{A}$ denotes the set of anchors and e_a denotes the entity behind the anchor a.

Alignment by Names of Entities (Wang et al., 2014a). For each triplet (h, r, t), h or t are replaced with their corresponding names, so we get (w_h, r, t), (h, r, w_t) and (w_h, r, w_t), where w_h denotes the name of h and w_t denotes the name of t.

$$\mathcal{L}_{AN} = \sum_{(h,r,t)} [\mathcal{L}_{triplet}(w_h, r, t) + \mathcal{L}_{triplet}(h, r, w_t) + \mathcal{L}_{triplet}(w_h, r, w_t)] \tag{9}$$

Alignment by Entities' Description (Zhong and Zhang, 2015). This alignment utilizes the co-occurrence of Wikipedia url and words in the description of that url page, which is similar to the PV-DBOW model in (Le and Mikolov, 2014).

$$\mathcal{L}_{AD} = \sum_{e\in\mathcal{E}} \sum_{w\in\mathcal{D}_e} [\log \Pr(e|w) + \log \Pr(w|e)] \tag{10}$$

where $\mathcal{D}_e$ denotes the description of an entity e. To clarify again, "url" is equivalent with "entity" in this paper. Combine these three kinds of alignment techniques, we get the whole alignment model:

$$\mathcal{L}_A = \mathcal{L}_{AA} + \mathcal{L}_{AN} + \mathcal{L}_{AD} \tag{11}$$

3.1.4 Url-Anchor Co-occurrence

For entity disambiguation, the entity relatedness graph is useful to capture the "topics" of an entity in Wikipedia. Thus we also hope to encode such information into our embedding. Specifically we further incorporate "url-anchor" co-occurrence to the training objective. "url" stands for the url of a Wikipedia page and "anchor" stands for the hyperlinks of anchor fields in that page.

$$\mathcal{L}_U = \sum_{e\in\mathcal{E}} \sum_{a\in\mathcal{A}_{\mathcal{D}_e}} [\Pr(e|e_a) + \Pr(e_a|e)] \tag{12}$$

where $\mathcal{A}_{\mathcal{D}_e}$ stands for all anchors in Wikpedia page $\mathcal{D}_e$. $\Pr(e|e_a)$ and $\Pr(e_a|e)$ are defined similarly as equation 6.

Considering knowledge model, text model, alignment model and url-anchor co-occurrence all together, we get the overall objective (likelihood) to maximize:

$$\mathcal{L} = \mathcal{L}_K + \mathcal{L}_T + \mathcal{L}_A + \mathcal{L}_U \tag{13}$$

3.1.5 Negative Sampling Refinement

In training phase, to avoid the computation of the normalizer in equation (2) and (6), we follow (Mikolov et al., 2013b) to transform the origin softmax-like objective to a simpler binary classification objective, which aims to distinguish observed data from noise.

To optimize for entity disambiguation, when using the context words to predict an anchor (entity), i.e., optimizing $\Pr(e_a|w)$, rather than uniformly sampling negatives from the vocabulary as (Mikolov et al., 2013b), we conduct our sampling according to the candidates' prior distribution.

3.2 Disambiguation Features Design

With the embeddings we train above, many entity disambiguation methods can directly take them as the words and entities' representation and redefine their features. In this section, we only design some simple features to illustrate the capability of the embeddings in disambiguation. In the section of experiment, we can observe that even a single embedding-based feature can beat the combination of several BoW-based features.

3.2.1 Mention-Entity Prior

This feature is directly counted from Wikipedia's anchor fields and measures the link probability of an entity e given a mention m. Prior is a strong indicator (Fader et al., 2009) to select the correct entity. However, it is unwise to take prior as a feature all the time because prior usually get a very large weight, which overfits the training data. Later in this paper, we will propose a classifier to tell when to use the prior or not.

3.2.2 Global Context Relatedness (E-GCR)

This feature comes from the hypothesis that the true entity of a mention will coincide with the meaning of most of the other words in the same document. So this feature sums up all idf-weighted relatedness scores between an entity candidate and each context word, then average them:

$$\forall e \in \Gamma(m), E-GCR(e,d|m) = \frac{1}{|d|}\sum_{w\in d}\mathrm{idf}(w)\cdot\Omega(e,w) \quad (14)$$

where $\Gamma(m)$ denotes the entity candidate set of mention m; d denotes the document containing m; $\Omega(e,w)$ denotes a distance-based relatedness $b-\frac{1}{2}\left\|\mathbf{e}-\mathbf{w}\right\|^2$, which is compatible with the embedding model.

3.2.3 Local Context Relatedness (E-LCR)

E-GCR can only coarsely rank topic-related candidates to one of the top positions. But sometimes there is nearly no relation between the true entity and the topic of the document:

Ex.1 "*Stefani, a DJ at The Zone 101.5 FM in Phoenix, **AZ**, sent me an awesome MP3 of the interview...*"

In this example, E-GCR will link ***AZ*** to ***AZ (rapper)*** because the context is all about music although ***Phoenix*** should be a strong hint to link ***AZ*** to ***Arizona***.

To avoid this kind of errors, we design a feature to describe the relatedness between an entity candidate and some important words around the mention. To identify these words, we turn to dependency parser provided by Stanford CoreNLP (Manning et al., 2014). Formulate this feature:

$$\forall e \in \Gamma(m), E-LCR(e,d|m) = \frac{1}{|S_{depend}|}\sum_{w\in S_{depend}}\Omega(e,w) \quad (15)$$

where S_{depend} is the set consisting all adjacent words of m in the dependency graph of the document d.

3.2.4 Local Entity Coherence (E-LEC)

In practice, there are usually many casual mentions linked to an entity, such as ***w v*** for ***West Virginia***.

Ex.2 "*We would like to welcome you to the official website for the city of **Chester**, w v.* "

In this case, "***w v***" should be a strong hint for the disambiguation of "***Chester***". However, "***w***", "***v***" or "***w v***" is too casual to catch useful information if we only take their lexical expression. So we should not only take the relative surface forms but also their entity candidates into consideration. Then the entity "***West Virginia***" will be quite helpful to link "***Chester***" to "***Chester, West Virginia***"

This feature is similar to the previous collective or topic-coherence methods. And our local entity coherence is more accurate because we only consider relative mentions/entities around rather than all entities in a document. Formulate this feature:

$$\forall e \in \Gamma(m), E-LEC(e,d|m) = \frac{1}{|S_{depend}|}\sum_{w\in S_{depend}}\max_{e'\in\Gamma(w),e'\neq e}\Omega(e,e') \quad (16)$$

3.3 Two-layer Disambiguation Model

To balance the usage of prior and other features, we propose a two-layer disambiguation model. It includes two steps: (1) Build a binary classifier to give a probability p_{conf} denoting the confidence to use prior only. Features used to construct this classifier are E-GCR, mention word itself and context words in a window sized 4 around the mention. (2) If p_{conf} achieve a designated threshold ξ, we only adopt prior to select the best candidate, otherwise we only consider other embedding-based features described in section 3.2. Formulate this model:

$$\forall m, e^* = \begin{cases} \arg\max_{e\in\Gamma(m)} prior(e|m), & p_{conf} \geq \xi \\ \arg\max_{e\in\Gamma(m)} \sum_i^{|F|} w_i \cdot f_i, & p_{conf} < \xi \end{cases} \quad (17)$$

where e^* is the entity we choose for the mention m.

4 Experiments

In the experiments, we first compare our embedding-based features with some traditional BoW-based features. Then we illustrate the capability of the two-layer disambiguation model. After that we compare our embedding technique EDKate in entity disambiguation tasks with some other straightforward work-arounds. Finally we incorporate mention detection and construct a disambiguation system to compare with other existing systems.

4.1 Data

We take Freebase as the KB, full Wikipedia corpus as text corpus. For comparison, we also use some small benchmark corpus for testing purpose.

4.1.1 Wikipedia

We adopt the Wikipedia dumped from Feb. 13th, 2015. With the raw htmls, we first filter out non-English and non-entity pages. Then we extract text and anchors information according to the html templates. After the preprocessing procedures, we get 4,532,397 pages with 93,299,855 anchors. Furthermore, we split the remained pages into training, developing and testing sets with proportion 8:1:1. In some experiments, only "valid entities" will be considered and a "(filtered)" tag will be added to the name of the dataset. For statistical summary, please refer to Table 1.

4.1.2 Valid Entities

In some experiments, we limit our KB entities to the Wikipedia training set and remove entities which are mentioned less than 3 times in Wikipedia training set for efficiency. We call the remaining entities "valid entities".

4.1.3 Knowledge Base

We use Freebase dumped from Feb. 13th, 2015 as our knowledge base. We only want to link mentions to Wikipedia entities so we filter out triplets whose head or tail entity isn't covered by Wikipedia. Finally we get 99,980,159 triplets. If we only consider valid entities, there are 37,606,158.

4.1.4 Small Benchmark Corpus

Besides Wikipedia, we also evaluate our embedding-based method in some small benchmark datasets. KBP 2010 comes from the KBP's annual tracks held by TAC and contains only one mention in one document. AQUAINT is originally collected by (Milne and Witten, 2008) and mimics the structure of Wikipedia. MSNBC is taken from (Cucerzan, 2007) and focuses on news wire text; ACE is collected by (Ratinov et al., 2011) from the ACE co-reference dataset. For statistics in detail, see Table 1.

4.1.5 Difficult Set

We find that in all the data sets, large part of the examples can be simply well solved by the mention-entity prior without considering any contexts. But there indeed exist some examples the prior cannot work well. We think disambiguation should pay more attention to this part of examples rather than the part where prior already works well. Thus from the testing sets "Wikipedia:test (filtered)" and "KBP 2010 (filtered)", we collect the cases where prior cannot rank the correct entity to top 1 and construct the separate "difficult" set.

Dataset	# documents	# mentions
Wiki:all	4,532,397	93,299,855
Wiki:all (filtered)	2,476,438	52,422,949
Wiki:train (filtered)	1,567,080	37,956,309
Wiki:develop (filtered)	454,906	7,248,850
Wiki:test (filtered)	454,452	7,217,790
Wiki:test (filtered, difficult)	454,452	1,069,428
KBP 2010	1020	1020
KBP 2010 (filtered)	780	780
KBP 2010 (filtered, difficult)	780	183
AQUAINT	50	727
ACE	35	257
MSNBC	20	747

Table 1: Statistics for each corpus

4.2 Embedding Training

We use stochastic gradient descent (SGD) to optimize the objective (see equation (13)). We set the dimension of word and entity embeddings to 150 and initialize each element of an embedding with a random number near 0. For the constant b, we empirically set it to 7.

In knowledge model, we use Freebase as our knowledge base. We don't set a fix epoch number and the knowledge training thread will not terminate until the text training thread stop. Furthermore, we also adapt the learning rate in knowledge training to that in text training. When a triplet (h, r, t) is considered, the numbers of negative samples to construct $(\tilde{h}, r, t)$, $(h, \tilde{r}, t)$ and $(h, r, \tilde{t})$ are all 10, in which $\tilde{h}$ and $\tilde{t}$ are uniformly sampled from $\mathcal{E}$ while $\tilde{r}$ is uniformly sampled from $\mathcal{R}$.

In text model, we use the filtered Wikipedia training set as our text corpus. We set the number of epoch to 6 and set initial learning rate to 0.025, which will decreases linearly with the training process. When a word is encountered, we take words inside a 5-word-window as co-occurred words. For each co-occurred word, we sample 30 negatives from the unigram distribution raised to the 3/4rd power.

In alignment model, "alignment by Wikipedia anchors" and "alignment by entity names" can be absorbed into text model and knowledge model re-

spectively. For "alignment by entity's description, we sample 10 negatives in $\Pr(e|w)$ and 30 negatives in $\Pr(w|e)$.

For $\Pr(e_a|w)$ in "url-anchor co-occurrence", we sample 20 negatives from the candidate list of the anchor mention and 10 negatives from the whole entity set.

To balance the training process, we give knowledge model 10 threads and text model 20 threads. We adopt the share-memory scheme like (Bordes et al., 2013) and don't apply locks.

4.3 Comparison between Embedding-based and BoW-based Feature

We set up this experiment to exhibit the expressiveness of our embeddings. We compare E-GCR (global context relatedness) with some traditional BoW-based features. Moreover, in this experiment, we report the results on "difficult set" where the mention-entity prior fails. Following the same metric used in (Cucerzan, 2011), we take accuracy to evaluate the disambiguation performance, that is, the fraction of all mentions for which the correct entity is ranked to top 1.

4.3.1 Implementation

For embedding-based features, we only consider the E-GCR, which is more comparable with the BoW-based features B-CS and B-TS we use here because they all consider the whole document as context. These BoW-based features include: (1)Mention-Entity Prior; (2)BoW Context Similarity (B-CS). This feature is proposed by (Cucerzan, 2011). First, for each entity in Wikipedia, take all surface forms of anchors in that page as its representation vector. Then compute scalar product between this representation vector and the context word vector of a given mention; (3)BoW Topic Similarity (B-TS). First construct the topic vector for each entity from category boxes like (Cucerzan, 2011). Then compute scalar product between topic vector and context word vector of a given mention.

4.3.2 Results

From Table 2, we get (1) E-GCR can beat the combination of several BoW-based features. This is mainly because, embeddings training owns a inner optimizaiton objective and embraces the information of these BoW-based representations. (2) Embedding-based feature appear robust and significantly outperform BoW-based features in difficult set, which indicate that these BoW representation cannot well catch the information in these cases while embeddings are still expressive . (3) Unlike the situation in difficult set, the gap between "E-GCR" and "Prior+B-CS+B-TS" is not so large mainly because "difficult set" only occupy a small proportion and prior would cover the drawbacks of BoW-based features.

This experiment hints us to pay more attention to the difficult set, which is helpful to improve the overall performance.

Feature	Wiki:test (filtered)		KBP 2010 (filtered)	
	overall	difficult	overall	difficult
Prior	0.8488	0	0.7645	0
Prior+B-CS	0.8545	0.0587	0.7645	0.0308
Prior+B-CS+B-TS	0.8680	0.1609	0.7907	0.1437
B-CS	0.6375	0.3159	0.3648	0.3210
B-CS+B-TS	0.6793	0.3943	0.5422	0.4491
E-GCR	**0.8738**	**0.5183**	**0.8445**	**0.6358**

Table 2: Comparison between Embedding-based Feature and BoW-based Feature

Model Type	Wiki:test (filtered)		KBP 2010 (filtered)	
	overall	difficult	overall	difficult
Linear	0.8671	0.1310	0.7791	0.0617
Two-layer	**0.8931**	**0.4795**	**0.8474**	**0.5140**

Table 3: Comparison between Linear Disambiguation Model and Two-layer Model

4.4 Comparison between Linear and Two-layer Disambiguation Model

In this section we evaluate the quality of the two-layer disambiguation model and compare it with the linear disambiguation model (Cucerzan, 2011). Moreover, we also report results in the "difficult set" defined above to see whether our two-layer model could balance prior and other features or not. The features we use here are prior and E-GCR. Accuracy is used as the evaluation metric.

4.4.1 Implementation

We use logistic regression for both models. For the two-layer model, we first apply the prior classification and get p_{conf}. Here we set the threshold ξ to 0.95 according to experiments in development set.

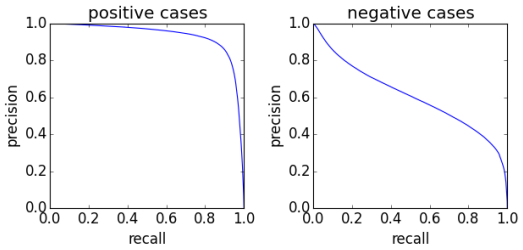

Figure 1: Precision-Recall Curves for Prior Classification

4.4.2 Results

From Figure 1, we see that the classifier is quite good at classifying positive cases to the correct class. From Table 3, we observe that our two-layer model receive a promising result in overall and difficult set against the linear model. This evidence indicates the prior classifier works and the two-layer model can balance the usage of prior and other features.

4.5 Comparison between Different Embeddings

This experiment compares our embeddings technique (EDKate) with some other methods: (Mikolov et al., 2013b), (Wang et al., 2014a) and (Zhong and Zhang, 2015). Here, We only consider the cases given mentions and take accuracy as the evaluation metric.

4.5.1 Implementation

For (Mikolov et al., 2013b), we directly take their word2vec model and replace anchor surface with the entity symbol in the training corpus. In this way, we get embeddings for words and entities. For (Wang et al., 2014a) and (Zhong and Zhang, 2015), we completely follow the model in the original paper. For our method. We use the knowledge model and text model described in (Wang et al., 2014a) and combine the alignment techniques in both of (Wang et al., 2014a) and (Zhong and Zhang, 2015). Moreover, we add "url-anchor" co-occurence to the training objective and refine the negative sampling method by having entities in candidates list more probable to be sampled. In this experiment, we only use E-GCR as our feature for simplicity.

4.5.2 Results

From Table 4, we observe that (Wang et al., 2014a) outperform (Mikolov et al., 2013b), which indicates the introduction of some structure information like knowledge base is quite beneficial. And the utilization of description message for entities improve the performance as well. For our method EDKate, we further take advantages of "url-anchor" co-occurrence and special sampling method, which make embeddings more expressive and guarantee the performance.

Embedding Version	Wiki: test (filtered)	KBP 2010 (filtered)
(Mikolov et al., 2013b)	0.8062	0.7311
(Wang et al., 2014a)	0.8283	0.7922
(Zhong and Zhang, 2015)	0.8355	0.7965
EDKate	**0.8738**	**0.8445**

Table 4: Comparison between Different Embeddings

Method	Accuracy on KBP 2010
(Lehmann et al., 2010)	0.806
(He et al., 2013)	0.809
(Sun et al., 2015)	0.839
(Cucerzan, 2011)	0.873
EDKate	**0.889**

Table 5: Comparison with other reported results on KBP 2010

4.6 Comparison with Reported Results on KBP 2010

In this section, we will compare our result on KBP 2010 with other existing reported results, in which Cucerzan (2011) holds the best record in KBP 2010 so far; Lehmann et al. (2010) rank first in the 2010 competition while He et al. (2013) and Sun et al. (2015) adopt neural-network-based methods. We still use accuracy as the evaluation metric because KBP 2010 specifies the input mentions. Because some papers only report the accuracy with 3 decimal places, we unify all results to 3 decimal places.

4.6.1 Implementation

We take the dataset "Wikipedia:all" to train embeddings here and use all features we defined in section 3.2. In this experiment, we adopt the unfiltered version of KBP 2010 as the test corpus.

4.6.2 Results

Table 5 shows that EDKate outperforms the current best record (Cucerzan, 2011) in KBP 2010 dataset. Sun et al. (2015) apply convolution neural network and take advantages of (Mikolov et al., 2013a)'s word2vec as input but it seems not so ef-

System	AQUAINT	ACE	MSNBC
WikipediaMiner	0.8361	0.7276	0.6849
Wikifier_v1	0.8394	0.7725	0.7488
EDKate	0.8515	0.8079	0.7550
Wikifier_v2	**0.8888**	**0.8530**	**0.8120**

Table 6: Comparison with other Wikification systems in BoT F1 metric

fective as ours, which shows the importance of the embedding quality in this disambiguation task.

4.7 Comparison with Other Wikification Systems

In this section, we equip EDKate with mention detection and compare our system with WikipediaMiner (Milne and Witten, 2008), Wikifier_v1 (Ratinov et al., 2011) and Wikifier_v2 (Cheng and Roth, 2013). For the evaluation metric, we adopt the Bag-of-Title (BoT) F1 evaluation metric which is used in all other systems we choose here.

4.7.1 Implementation

We first make use of all mentions in the mention-entity table to construct a Trie-tree, which is used to detect mentions in input text. To remove noise, we simply retain mentions which contain at least one noun and filter mentions that completely consist of stop words. Then we apply our disambiguation technique to the mentions detected. The same as experiment 4.6, we make use of all features described in section 3.2 here.

4.7.2 Results

Table 6 shows that our embedding-based method EDKate is better than two popular systems but cannot outperform Wikifier_v2 in these three datasets. It should be mentioned that Wikifier_v2 is largely based on Wikifier_v1 and its magic is to add relational inference with some handcrafted rules. Actually, the embedding methods can performs well to model relations (Wang et al., 2014a), so the idea to introduce relational information into our current framework is promising and will be the future work.

5 Conclusion

In this paper, we propose to refine a knowledge and text joint learning framework for entity disambiguation tasks and learn semantics-rich embeddings for words and entities. Then we design some simple embedding-based features and build a two-layer disambiguation model. Extensive experiments show that our embeddings are very expressive and is quite helpful in the entity disambiguation tasks.

References

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

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Razvan C Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of European Chapter of the Association for Computational Linguistics*, volume 6, pages 9–16.

Taylor Cassidy, Zheng Chen, Javier Artiles, Heng Ji, Hongbo Deng, Lev-Arie Ratinov, Jing Zheng, Jiawei Han, and Dan Roth. 2011. Cuny-uiuc-sri tac-kbp2011 entity linking system description. In *Proceedings of Text Analysis Conference*.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

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

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, volume 7, pages 708–716.

Silviu Cucerzan. 2011. Tac entity linking by performing full-document entity extraction and disambiguation. In *Proceedings of Text Analysis Conference*, volume 2011.

Anthony Fader, Stephen Soderland, Oren Etzioni, and Turing Center. 2009. Scaling wikipedia-based named entity disambiguation to arbitrary web text. In *Proceedings of the International Joint Conferences on Artificial Intelligence Workshop on User-contributed Knowledge and Artificial Intelligence: An Evolving Synergy, Pasadena, CA, USA*, pages 21–26.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *ACL (2)*, 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*, pages 782–792.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of Special Interest Group on Knowledge Discovery and data Mining*, pages 457–466.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

John Lehmann, Sean Monahan, Luke Nezda, Arnold Jung, and Ying Shi. 2010. Lcc approaches to knowledge base population at tac 2010. In *Proceedings of Text Analysis Conference*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of Association for Computational Linguistics*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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*, pages 3111–3119.

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of Information and knowledge management*, pages 509–518.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of Association for Computational Linguistics*, pages 1375–1384.

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*, 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. 2014a. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics*, pages 1591–1601.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of Association for the Advancement of Artificial Intelligence*, pages 1112–1119.

Huaping Zhong and Jianwen Zhang. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.