A Recurrent Model for Collective Entity Linking with Adaptive Features*

Xiaoling Zhou,[1] Yukai Miao,[1] Wei Wang,[1,2] Jianbin Qin[3]
[1]School of Computer Science and Engineering, UNSW, Australia
[2]College of Computer Science and Technology, DGUT, China
[3]Shenzhen Institute of Computer Science, Shenzhen University, China
xiaolingz@cse.unsw.edu.au, {yukai.miao, weiw}@unsw.edu.au, jqin@sics.ac.cn

Abstract

The vast amount of web data enables us to build knowledge bases with unprecedented quality and coverage. Named Entity Disambiguation (NED) is an important task that automatically resolves ambiguous mentions in free text to correct target entries in the knowledge base. Traditional machine learning based methods for NED were outperformed and made obsolete by the state-of-the-art deep learning based models. However, deep learning models are more complex, requiring large amount of training data and lengthy training and parameter tuning time.

In this paper, we revisit traditional machine learning techniques and propose a light-weight, tuneable and time-efficient method *without* using deep learning or deep learning generated features. We propose novel adaptive features that focus on extracting discriminative features to better model similarities between candidate entities and the mention's context. We learn a local ranking model based on traditional and the new adaptive features based on the learning-to-rank framework. While arriving at linking decisions individually via the local model, our method also takes into consideration the correlation between decisions by running multiple recurrent global models, which can be deemed as a learned local search method. Our method attains performances comparable to the state-of-the-art deep learning-based methods on NED benchmark datasets while being significantly faster to train.

1 Introduction

Thanks to the huge amount of data on the Web, we are able to automatically create knowledge bases with high quality and coverage, such as YAGO (Suchanek, Kasneci, and Weikum 2008), and Knowledge Vault (Dong et al. 2014). A crucial step in building a knowledge base is to deal with the natural language ambiguity. For example, the same string, or *mention*, can refer to different entities in different documents — *New York* can be the state in U.S. , or the film directed by Kabir Khan in 2009, among other possibilities.

The task of eliminating this kind of ambiguity and linking the entity mention in text to its targeted entity entry in a knowledge base is called Named Entity Disambiguation (NED), or Entity Linking. It not only enables us to build high-quality knowledge bases, but also aids information retrieval, information extraction and many other kinds of artificial intelligence applications such as reasoning and question answering (Kataria et al. 2011).

NED is a challenging problem due to *entity ambiguity* and *mention variations*. As mentioned before, a mention *New York* can refer to a state as well as a film entity. Likewise, the city *New York* can be referred to by various mentions such as *New York*, *New York City*, *Big Apple* or *NY*.

Traditional machine learning techniques (Hoffart et al. 2011; Lazic et al. 2015; Globerson et al. 2016) were first employed to solve the NED task. They are based on well-understood machine learning models (e.g., SVM (Ratinov et al. 2011)) and can be trained efficiently on commodity computers. Recently, deep learning based methods (Yamada et al. 2016; Gupta, Singh, and Roth 2017; Ganea and Hofmann 2017; Le and Titov 2018) become the prevalent solution, as they outperform machine learning based methods. The performance advantages are due to the features learned by powerful deep learning models (e.g., entity embeddings learned from the entire Wikipedia corpus). Nevertheless, deep learning also brings about several weaknesses: (1) the models are large and complex; (2) they require large amount of training data, hence incurring high model training and parameter tuning costs even with hardware accelerations (e.g., GPUs).

After comparing these two paradigms, we wonder *whether we can have a traditional machine learning-based method that enjoys state-of-the-art performances*. We answer in the affirmative, and in this paper, we propose a simple yet highly effective model that does not use deep learning models or features. As such, our model offers several unique advantages: it is lightweight both in terms of the model size and the (re-)training time (even without GPUs), easy to fine-tune and incorporate existing and new engineered features, and easy to deploy and customize in many real application scenarios.

Our model consists of a **local model** and *a series of* **global models**. Given a mention m, the local model uses a Learning-to-Rank model trained on traditional and novel *adaptive* local features to compute the ordered candidate entities. The (first) global model takes into consideration the ranked lists of other mentions in the same document by computing certain coherence features from top-ranked entities of the other mentions, and reranks the candidates for mentions

*X. Zhou and J. Qin are the co-corresponding authors.

in query. The reranking of the candidates for every mention will potentially invalidate the coherence features computed for some entities. Therefore, we propose to train global models *recurrently*: feeding the reranked candidate lists of every mention obtained from the i-th global model as the input to the $(i+1)$-th global model. Typically only a few recurrent global models are needed for the final global model to obtain the best performance. Experimental results over the NED benchmark datasets show that our method achieves performances comparable to those of the state-of-the-art models, while significantly faster to train.

2 Problem Definition

Denote the query document as $Q = [w_1, w_2, \ldots, w_{|Q|}]$, and there are α mention spans, $\{ m_i \}_{i=1}^{\alpha}$. We have a collection of entities $E = \{ e_i \}_{i=1}^{n}$ in the knowledge base, the NED task is to map each mention m_i to the targeted entity e_j in E.

Figure 1 is an illustration of the NED task. The text in blue box is query document, and the underlined text spans are *mention*s that need to be resolved. The mentions are linked to correct entities stored in the knowledge base (shown in the ellipse).

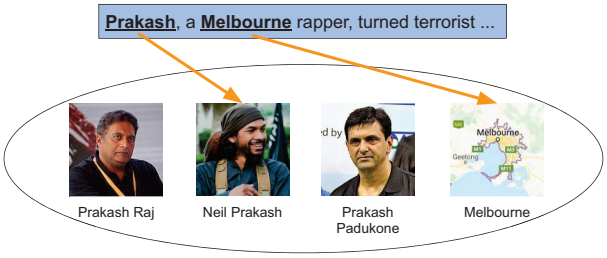

Figure 1: An Example of the NED Task

3 Related Work

Entity linking has been widely explored in recent years. Usually, an entity linking system consists of two components: candidate entity generation and referent entity prediction (Shen, Wang, and Han 2015).

Candidate entity generation Most of the work uses a mention-entity dictionary for generating the candidate entities, and the dictionary is built by utilizing Wikipedia hyperlinks (Barrena, Soroa, and Agirre 2015; 2016; He et al. 2013). Hoffart et al. (Hoffart et al. 2011) explored external information from other knowledge bases for retrieving candidate entity, such as utilizing *sameAS* relation in DBpedia, and *means* relation in YAGO. To improve the coverage of the generated candidate entities list, Han and Zhao (Han and Zhao 1999) employed the Multi-way entity candidate detection module to detect possible entities.

Referent entity prediction Ling et al. (Ling, Singh, and Weld 2015) proposed a modular, unsupervised, end-to-end entity liking system, VINCULUM, which ranks candidate entities by sum of feature scores from mention and entity. Contextual information always conveys important information for entity linking, thus Pan et al. (Pan et al. 2015) adopted the Abstract Meaning Representation (Banarescu et al. 2013) technique to model the context and utilized Jaccard similarity measurement to calculate the mention-entity similarity. There are also methods (Barrena, Soroa, and Agirre 2016; Yamada et al. 2016; Ganea and Hofmann 2017) that use word-embeddings to represent the contexts of mention and candidate entities, and use cosine distance as the similarity measurement.

Some graphical model based methods (Hoffart et al. 2011; Barrena, Soroa, and Agirre 2015; 2016; Nguyen, Theobald, and Weikum 2016) are also proposed. Hoffart et al. (Hoffart et al. 2011) constructed a weighted and undirected graph, where mentions and their candidate entities were represented as nodes. Then an iterative heuristic is used to remove unpromising edges and finally find the correct linkings. Barrena et al. (Barrena, Soroa, and Agirre 2015) proposed a Bayesian network based method, and Barrena et al. (Barrena, Soroa, and Agirre 2016) extended the previous work by introducing new evidences to the Bayesian network, such as distributional similarity and selectional preferences. Nguyen et al. (Nguyen, Theobald, and Weikum 2016) presented J-NERD to perform NER and NED jointly by means of a probabilistic graphical model.

He et al. (He et al. 2013) proposed one of the earliest deep neural network (DNN) based framework for NED task. Later, Sun et al. (Sun et al. 2015) introduced a new neural network approach which simultaneously takes into consideration the mention as well as its context; convolutional neural network (CNN) (Kalchbrenner, Grefenstette, and Blunsom 2014) was employed to produce fixed-length vectors for contexts. One common problem with (He et al. 2013) and (Sun et al. 2015) is that they only modelled the context information and did not consider the topical coherence of entities linked to the mentions in the query document. As a complement, Huang et al. (Huang, Heck, and Ji 2015) presented a deep semantic relatedness model to model the topical coherence between entities.

Obtaining the best globally coherent solution is computationally prohibitive, therefore many approximate inference techniques are used. Hoffart et al. (Hoffart et al. 2011) used graph pruning. Cheng and Roth (Cheng and Roth 2013) adopted integer linear programming. Ratinov et al. (Ratinov et al. 2011) used ranking SVM. Ganea et al. (Ganea et al. 2016) and Ganea and Hofmann (Ganea and Hofmann 2017) used Belief Propagation and its variant Loopy Belief Propagation respectively. Globerson et al. (Globerson et al. 2016) performed a single round of message passing with attention mechanism. Cao et al. (Cao et al. 2018) applied Graph Convolutional Network to integrate global information. Le and Titov (Le and Titov 2018) extended the work of (Ganea and Hofmann 2017) by modelling latent relations between textual mentions. Xue et al. (Xue et al. 2019) proposed random walk on graph-based neural network model to infer semantic relations between entities.

4 Overview

We illustrate the architecture of our solution in Figure 2. Our model consists of a **local model** ($f^{(0)}$) and *a series of* **global**

models ($\{ f^{(i)} \}_{i=1}^{T}$), trained using the same ranking model to be presented in Section 5.2, but with different feature sets.

1. Given each mention in the query document, we search a *candidate entity dictionary* using the mention as the search key, thus obtain a set of k candidate entities that the mention may be linked to.
2. Then, we construct the local feature vector $\mathbf{x}_l$ for each entity and feed them into the local model to convert the set of candidate entities to a ranked list of candidate entities with decreasing probabilities of being the correct entities to be linked to. The local feature vector consists of a set of novel local *adaptive* features (detailed in Section 5.3) and features known to be effective for the NED task.
3. The (first) global model resolves mentions by considering not only local features of each candidate entity, but also the impacts of ranked list of candidates (the prediction result of local model) of other mentions in the same document, i.e., coherence features that capture the relatedness between current candidate entity and top-ranked entities of the other mentions. This is done via constructing the global feature vector, $\mathbf{x}_g$, by enhancing the local feature vector $\mathbf{x}_l$ with (aggregated) coherence features and training a new reranking model.
4. The reranking of the candidates for every mention will potentially invalidate the coherence features computed for some entities. Therefore, we re-train the global model iteratively by feeding the reranked candidate lists of every mention obtained from the i-th global model as the input to the $(i+1)$-th global model, so that the related entity contexts we use to compute coherence scores will be more and more accurate and closer to ground truth entities, thus improving the model accuracy. The iteration terminates when the performance stops increasing on the validation set.

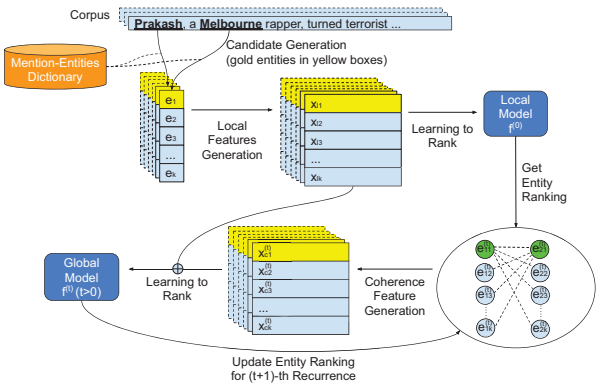

Figure 2: System Architecture

5 The Local Model

We first describe the candidate generation method, followed by the local model, and finally features employed in the local model.

5.1 Candidate Entity Generation

A candidate entity dictionary is the key data structure to provide candidate entities for mentions as well as related statistical information useful for the disambiguations. It is usually provided or constructed from the knowledge base for candidate entity generation.

Following previous work, we construct the mapping dictionary using Wikipedia hyperlinks, where the anchor text is treated as mention, and the targeted Wikipedia page title is added as a candidate entity of the mention, by crawling Wikipedia pages including Disambiguation pages and Redirect pages. For example, the anchor text "Washington" can refer to both the State in United States and the President of United States. Besides, we also add YAGO (Hoffart et al. 2011) dictionary following (Ganea and Hofmann 2017). In addition, we also store the Wikipedia page for every entity as its *description document*, which will be used later to capture the similarity with query documents.

We perform exact search in the dictionary using the mention as the key, and return up to k entities as candidate entities.

5.2 Local Model

At this stage, we cast the task into a ranking problem, and rank the unordered candidate entities based on the various local features, including novel *adaptive* features.

Learning to Rank The following section presents the ranking model, and the detailed features are described later.

Our ranking model is the standard learning-to-rank model with the pair-wise loss. Unlike previous models (Ratinov et al. 2011) that use SVM as the underlying classifier, we choose to use gradient boosting models (e.g., `xgboost`), which consider non-linear models and avoids at least $O(n^2)$ training cost.

The learning-to-rank (L2R) model takes as input l objects represented by the feature vector $\{ \mathbf{x}_i \}_{i=1}^{l}$, and ranks them into an ordered list $\hat{\mathcal{L}} = \{ \pi(\mathbf{x}_i) \}_{i=1}^{l}$, such that it minimizes some loss function between the $\hat{\mathcal{L}}$ and the ground truth list $\mathcal{L}$.

The model scores objects $s_i = F(\mathbf{x}_i)$ and $s_j = F(\mathbf{x}_j)$ and models the probability that $\mathbf{x}_i$ should be ranked higher than $\mathbf{x}_j$ via a sigmoid function:

$$P_{ij} \stackrel{\text{def}}{=} P(e_i \prec e_j) = \frac{1}{1 + e^{-\beta \cdot (s_i - s_j)}} \quad (1)$$

where the hyper-parameter β controls the shape of the sigmoid function. The final loss is based on cross entropy loss:

$$\mathbb{L} = \sum_{(e_i, e_j) \in I} -P_{ij}{}^{*} \log P_{ij} - (1 - P_{ij}{}^{*}) \log(1 - P_{ij}) \quad (2)$$

where $P_{ij}{}^{*}$ is the probability of ranking $\mathbf{x}_i$ before $\mathbf{x}_j$ in the ground truth and I contains all pairs of input objects with different ground truth rankings.

Since we only have the ground truth entity for each mention, to apply the L2R model to our problem, we only gen-

erate $2 \cdot (k-1)$ pairs as I in the following way:

$$\begin{cases} {P_{ij}}^* = 1 & \text{, if } i = 1 \wedge i \neq j \\ {P_{ij}}^* = 0 & \text{, if } i \neq 1 \wedge j = 1 \end{cases}$$

We use the gradient boosting model to learn the scoring function F.

5.3 Local Features

We introduce features used by the local model.

Prior Feature Two prior features are computed for each entity. Each candidate in our candidate entity dictionary corresponds to a unique Wikipedia page, hence we can exploit statistics of inter-page links. The first feature, $P(e \mid m)$, is the fraction of times entity e is the linked target page for mention m. This is a highly valuable feature for the disambiguation task: using this feature alone achieves a reasonable performance, which is presented in Section 7. The second feature, $P(e)$, is the fraction of links where e is the target page.

Adaptive Features Given the Wikipedia description page as the *entity context* and query document as the *mention context*, previous work computes the TF-IDF summaries of the two contexts, which are the bag-of-words representation of top-k tokens weighted by the TF-IDF formula, and then use the cosine similarity between the two summaries. Note that the TF-IDF values are calculated based on entire Wikipedia corpus.

$$\text{TF-IDF}(w, d, D) = \text{TF}(w, d) \cdot \text{IDF}(w, D) \tag{3}$$

Here, $\text{TF}(w, d)$ measures the number of times term w occurs in document d, and $\text{IDF}(w, D)$ measures the inverse of the fraction of documents d in corpus D that contains term w.

$$\text{IDF}(w, D) = \log \frac{|D|}{|\{\, d \mid w \in d, d \in D \,\}|} \tag{4}$$

As such, one shortcoming of the method is that it does not provide adaptive and discriminating similarity measurement for a given mention and its k candidate entities. For example, the term `USA` may appear frequently in the entire corpus, thus having a relatively low IDF value and a low probability to enter the summary. However, if only one of the candidate entities' description document contains the term `USA`, this term should be a highly discriminative feature and could aid in distinguishing entities that are linkable from the same mention but from different countries.

To overcome this problem, we first create the *adaptive corpus* of the mention that consists of the description documents associated with one of the k candidate entities for the current mention. We then compute two features for each term w based on this adaptive corpus. The first is the TF-IDF(w, d, D) scores, where the IDF(w) is computed based on the adaptive corpus instead of the entire corpus. The second feature is the distribution of TF(w, d, D) values in the adaptive corpus. We summarize the distribution using the entropy, normalized to $[0, 1]$ by $\log k$, and denote it by Entropy(w).

Let $e.D$ and $m.D$ denote the description document associate with entity e and the context association with m respectively, the new adaptive textual similarity between the mention m and e is computed as:

$$\psi_l(m, e) = \sum_{w \in m.D \cap e.D} \lambda(w) \cdot \text{TF}(w, m.D) \cdot \text{TF}(w, e.D)$$
$$\lambda(w) = \text{IDF}(w) \cdot (1 - \text{Entropy}(w))$$

In the above equation, we combine adaptive features IDF(w) and Entropy(w) to measure the power of a word w to discriminate all the candidate entities of m. Thus, a word with high IDF and low Entropy values is assigned with high weight (i.e., $\lambda(w)$). Finally, we compute the weighted inner product between the term frequency-weight vectors of the mention context ($m.D$) and the candidate entity's description ($e.D$).

String Similarity Feature To account for possible spelling variations, we also consider several similarity measures based on the mention string and entity string, including EditDistance(m, e), StartWith(m, e), and EndWith(m, e).

Coreference Features It is not uncommon for multiple mentions in the same document to refer to the same entity, but each time with a different mention string. For example, the full name of a person may appear at the beginning of a news report, but only the last name is used in the rest of the report when referring to the same person.

To exploit this phenomenon, we first use the off-the-shelf tool to perform coreference resolution to form a chain of coreferent mentions. Then we define the following coreference feature to boost our confidence of selecting candidate e as the linked result of m.

$$\psi_r(m, e) = \max_{m' \in \text{Chain}(m)} \{\, P(e \mid m')) \,\} \tag{5}$$

where Chain(m) is the chain of coreferent mentions.

6 Global Models

In our global model, we need to form global feature vectors and then use an independent L2R model (i.e., with its own parameters) to rerank the candidate list for each mention individually. In the following, we first describe the coherence feature we use, and then introduce the recurrent global model as another solution to the structured prediction problem.

6.1 Coherence Features

Coherence features capture the relatedness between targeted entities of mentions in the query document. An assumption of the global model is that the targeted (i.e. Ground Truth) entities of mentions belonging to the same document are correlated to some extent.

We consider two types of coherence features, based on known effective measurements: Normalised Google Distance (NGD) and Pointwise Mutual Information (PMI) (Ratinov et al. 2011), both based on the overlap in incoming/outcoming links of the two entities. Given a Wikipedia

entity collection W, entities e_1 and e_2 with a set of incoming/outcoming links L_1 and L_2 respectively, NGD and PMI are defined as follows:

$$\text{NGD}(e_i, e_2) = \frac{\log(\max(|L_1|, |L_2|)) - \log(|L_1 \cap L_2|)}{\log(|W|) - \log(\min(|L_1|, |L_2|))} \quad (6)$$

$$\text{PMI}(e_i, e_2) = \frac{(|L_1| \cap |L_2|)/|W|}{|L_1|/|W||L_2|/|W|} \quad (7)$$

6.2 Recurrent Solution

It is well-known that linking decisions of different mentions in the same query document affect each other. The best local decision of a mention m_1 may not be the best if one considers the linking decision of m_2. A popular approach is to model this as a structured prediction problem, i.e., to model $P(e_1, e_2, \ldots, e_\alpha \mid m_1, m_2, \ldots, m_\alpha)$. However, finding optimal solution is NP-hard in general due to the exponential search space. Existing work solves it by one of the following approaches: (1) making structure simplifying assumption so that the inference has polynomial time exact solution, (2) using heuristic local search or amortized search (Ma et al. 2019), (3) or leaving it to a deep neural network to find out high-quality global solutions (Xue et al. 2019).

We note that they are all approximation schemes and thus we propose another heuristic model: we avoid the costly global inference by recurrently stacking multiple global models aware of the coherent feature. In the $(t+1)$-th recurrence of our global model, we denote the model used in the previous round as $f^{(t)}$, which is the t-th global model ($t > 0$) or the local model ($t = 0$) (See Figure 2). We denote the j-th entity in the ranked list predicted by $f^{(t)}$ for mention m_i as $e^{(t)}_{i,j}$. Then the current global solution based on the last round model is made of the top-ranked entities: $\mathcal{E}^{(t)} = [e^{(t)}_{1,1}, e^{(t)}_{2,1}, \ldots, e^{(1)}_{\alpha,1}]$. With this current global solution fixed, we then can compute aggregated coherence feature values for each $e_{i,j}$ and form their global vectors in this round. The aggregated features measure the coherent values by taking two input functions ($\mathcal{G}$ and r) and returns

$$\mathcal{G}(\{\, r(e_{i,j}, e_{u,1}) \,\}_{u=1}^{\alpha})$$

where $\mathcal{G}$ is the aggregation function, and we consider both $\max$ and avg, and r is the coherence feature, and we consider both NGD and PMI. Therefore, this forms a coherence feature vector of length 4, denoted as $\mathbf{x}_c^{(t+1)}$. We concatenate it with the local feature vector to form the global feature vector:

$$\mathbf{x}_g^{(t+1)} = [\mathbf{x}_l; \mathbf{x}_c^{(t+1)}]$$

Now we can learn a new learning-to-rank model that takes $\mathbf{x}_g^{(t+1)}$ as the input and produces a potentially different ranking for the k candidate entities. This newly learned model is the $(t+1)$-th global model. We keep on building such recurrent global models until its performance on the validation dataset starts to drop.

Our recurrent solution can be viewed as a kind of local search, as we evolve from one global solution $\mathcal{E}^{(t)}$ to another $\mathcal{E}^{(t+1)}$ iteratively. Yet it has important differences from existing heuristic-based local search (e.g., (Ma et al. 2019)) in: (1) *the transition strategy*: We learn the transition strategy that directly optimizes for the matching of ground truth, as compared with a blindly random transition or via a learned strategy optimized for some alternative goals. We also only require a small number of transitions instead of potentially many transitions in traditional local search-based method. (2) *the dynamic nature of the feature vectors*: The feature vectors used in our method are dynamic (i.e., the $\mathbf{x}_c^{(t)}$ part).

7 Experiment

In this section, we evaluate the performance of our proposed model on the most popular benchmark datasets for NED, and compare it with several previous state-of-the-art NED systems. [1]

7.1 Experiment Setting

Dataset We validate our models on six popular benchmark datasets for NED task used by previous works. The statistics are shown in Table 1.

- AIDA-CoNLL (Hoffart et al. 2011), a manually annotated benchmark dataset for NED task. It contains the news corpus from Reuters. The dataset is further split into training (AIDA-train), validation (AIDA-A), and test sets (AIDA-B).
- MSNBC (MSB), AQUAINT (AQU) and ACE2004 (ACE) are three datasets for Wikification, and are cleaned and updated by (Guo and Barbosa 2018).
- CWEB and WIKI are automatically extracted from ClueWeb and Wikipedia corpora by (Guo and Barbosa 2018; Gabrilovich, Ringgaard, and Subramanya 2013).

Table 1: Statistics of NED Datasets

Dataset	# Mention	#Doc	# Mention per doc
AIDA-train	18,541	946	19.5
AIDA-A(valid)	4,791	216	22.1
AIDA-B(test)	4,485	231	19.4
MSB	656	20	32.8
AQU	727	50	14.5
ACE	257	36	7.1
CWEB	11,154	320	34.8
WIKI	6,821	320	21.3

Competitive Models We name our model, RMA, and compare it with the following methods:[2]

Machine Learning based:

- (Hoffart et al. 2011; Ganea et al. 2016; Guo and Barbosa 2018) proposed graphical models with the combination of lexical and statistical features.

[1] Our code is released at https://github.com/tjumyk/RMA

[2] Note that we did not compare with recent work (Xue et al. 2019), as the mention numbers in their work do not agree with those reported in other work/datasets, which may affect evaluation scores significantly.

- (Lazic et al. 2015) proposed a probabilistic selective context model to deal with noisy and irrelevant contexts.
- (Chisholm and Hachey 2015) suggested to mine information from web links to help with NED.

Deep Learning based:

- (Huang, Heck, and Ji 2015) used deep neural network to measure entity semantic relatedness for topical coherence modeling.
- (Francis-Landau, Durrett, and Klein 2016) adopted convolutional neural network to model the semantic similarity between mention and entity context.
- (Globerson et al. 2016) introduced a coherence model with multi-focal attention mechanism.
- (Yamada et al. 2016) designed an embedding model specifically for NED task.
- (Ganea and Hofmann 2017; Le and Titov 2018) adopted deep learning model that combines entity embeddings and contextual attention mechanism.
- (Kolitsas, Ganea, and Hofmann 2018) proposed neural entity linking model using word embeddings.

Evaluation Metrics

We use the in-KB accuracy and Micro-F1 score (averaged across mention) as the evaluation metrics.

Note that an important factor to the performance of the model is the knowledge base used to generate candidate entities. Given a mention m, there are three cases:

1. m does not generate any candidate entities against the knowledge base.
2. m generates candidate entities, but they do not include the ground truth entity.
3. m generates candidate entities, and they contain the group truth entity.

The first two cases definitely leads to errors and hence will adversely impact the recall.

$$\text{Recall} = \text{Accuracy} = \frac{\text{correct\_linked}}{\text{total\_mentions}}$$

$$\text{Precision} = \frac{\text{correct\_linked}}{\text{processed\_mentions}}$$

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

In the above formula, 'total mentions' corresponds to the number of mentions reported in Table 1, and 'processed mentions' corresponds to the total mentions excluding m in case 1.

Training Details and Hyper Parameters Following previous work, we train our model on AIDA-train set, tune hyperparameters on AIDA-A set, and test on AIDA-B set (in-domain test) and other datasets (out-domain test). In candidate entity generation, we select top-50 candidate entities from the dictionary according to the entity prior. We adopt XGBoosting with *rank:pairwise* objective, and set the $n\_estimators$ to 4900, and max_depth to 6 according to the parameter tuning on AIDA-A set, and the iteration number T for global model is 4.

7.2 Overall Results

Table 2 shows the in-KB accuracy of our model compared with previous works on AIDA-B test set. The numbers for other methods are obtained from the original papers.

It can be seen that our model performs much better than previous (traditional) machine learning based methods (Hoffart et al. 2011; Chisholm and Hachey 2015; Ganea et al. 2016; Guo and Barbosa 2018) and is comparable with the state-of-the-art deep learning based (Yamada et al. 2016; Ganea and Hofmann 2017; Le and Titov 2018) models. Note that the numbers for (Huang, Heck, and Ji 2015), (Kolitsas, Ganea, and Hofmann 2018) and (Le and Titov 2018) are Micro-F1 scores reported in their work, which are usually higher than in-KB accuracy under current problem setting (referring to the evaluation metrics in Section 7.1).

Table 2: In-KB Accuracy on AIDA-B (In-domain Test)

Category	Methods	AIDA-B
Deep Learning	(Huang, Heck, and Ji 2015)	†86.6
	(Francis-Landau, Durrett, and Klein 2016)	86.9
	(Globerson et al. 2016)	91.0
	(Yamada et al. 2016)	91.5
	(Ganea and Hofmann 2017)	92.2
	(Kolitsas, Ganea, and Hofmann 2018)	†82.4
	(Le and Titov 2018)	†**93.1**
Machine Learning	(Hoffart et al. 2011)	78.0
	(Lazic et al. 2015)	86.4
	(Chisholm and Hachey 2015)	88.7
	(Ganea et al. 2016)	87.6
	(Guo and Barbosa 2018)	89.0
	RMA	**91.5**

Note: The number after † is the reported Micro-F1, as Accuracy was not reported in the paper.

Table 3: Micro F1 on Other Datasets (Out-domain Test)

Model	MSB	AQU	ACE	CWEB	WIKI
(Hoffart et al. 2011)	79.0	56.0	80.0	58.6	63.0
(Han, Sun, and Zhao 2011)	88.0	79.0	73.0	61.0	78.0
(Ratinov et al. 2011)	75.0	83.0	82.0	56.2	67.2
(Cheng and Roth 2013)	90.0	88.0	87.0	67.5	73.4
(Ganea and Hofmann 2017)	*93.7*	**88.5**	88.5	*77.9*	77.5
(Guo and Barbosa 2018)	92.0	87.0	88.0	77.0	**84.5**
(Le and Titov 2018)	**93.9**	*88.3*	**89.9**	77.5	78.0
RMA	93.2	*88.3*	*89.3*	**79.3**	*82.2*

Notes: The best ones are shown in **bold**, and the 2nd best in *italic*.

Table 4: In-KB Accuracy on AIDA-B (Ablation Test)

Models	AIDA-B
Final recurrent global model	91.5
Initial global model	89.8
Initial global model w/o adaptive features	87.5
Local model w/o coherence features	85.7
Prior	68.2

Table 3 shows the Micro-F1 scores for cross-domain test. The numbers in first 4 rows are obtained from (Guo and Barbosa 2018), and the rest is from original papers respectively.

It can be seen that on each dataset, the performance of our model is either the best or very close to the best scores

Table 5: Example of Distinctive Context Words Selected by Local Adaptive Features

Mention	Gold Entity	$P(e\|m)$ of Gold Entity	Prior Rank in Candidates	Distinctive Context Words Selected by Local Adaptive Features
Australia	Australia_Davis_Cup_team	0.002	12	Davis Cup, Mark Philippoussis, Roche, win
Bally	Bally_Shoe	0.092	4	shoe, Pairs, factory
Fox	Liam_Fox	0.006	15	Secretary, government, Foreign,the House of Commons, State, minister

Table 6: Example of Self-Correction Ability of Recurrent Global Model

Mention	Leipzig	Munich	Hamburg
Gold Entity	Leipzig/Halle_Airport	Munich_Airport	Hamburg_Airport
Prediction of local model	Leipzig	Munich	Hamburg_Airport
Prediction of initial global model	Leipzig	Munich_Airport	Hamburg_Airport
Prediction of iterated global model	LeipzigLeipzig/Halle_Airport	Munich_Airport	Hamburg_Airport

among all the previous models, which demonstrates the superior performance and generalisation ability of our model.

7.3 Ablation Test

Table 4 shows the ablation study of our proposed model on AIDA-B test set. The results show that iteration improves 1.7% of accuracy over the initial global model. The novel designed local features altogether improve 17.5% of accuracy over the model with prior only, and the local adaptive features improve the performance by 2.3% with the presence of coherence features. These numbers confirm the effectiveness of our recurrent model and carefully designed statistical local features.

7.4 Training Efficiency

Compared with the recent deep learning based models, the obvious advantages of our model are efficient training and testing, and easy adaptation to incorporate new features.

Our model takes 10 minutes for training a local model or 15 minutes for training a global model on AIDA-train with a 10-core 3.3GHz CPU, which compares favourably with SOTA deep neural based methods, e.g. (Le and Titov 2018) takes 1.5 hours to train on the same dataset with a single Titan X GPU and (Ganea and Hofmann 2017) needs 16 hours in the same setting. In addition, our testing on AIDA-B only takes 3 seconds.

7.5 Qualitative Analysis

Table 5 shows some examples of hard cases (where the mention prior of gold entity is low) that can be correctly resolved by our local model, together with the context words assigned with high weights according to the local corpus adaptive features. It can be seen that our local adaptive features can actually help to find highly distinctive words that are useful in identifying target entities.

Table 6 shows an example query that contains several airport mentions. In our local model, only one of them (Hamburg) is predicted correctly. With incorporation of global coherence features, another mention (i.e. Munich) is predicted correctly in our global model, and with the correction of disambiguation context using the initial global model results, the coherence feature of gold entity for the previous wrongly linked mention (Leipzig) is enhanced, and thus correctly linked by the newly re-trained global model.

7.6 Error Analysis

The errors in our model can be categorized into the following types. (1) The gold entity is inaccurate due to human error in manually annotated datasets or noisy data in automatically extracted datasets, while our model may have a more meaningful prediction. (2) The gold entity does not exist in the candidate entity list produced by the dictionary (3) The gold entity is in the candidate entity list but its prior is out of the top-50 and thus discarded by our model. As shown in Table 7, we have far more mentions with extremely low prior on the gold entity than (Ganea and Hofmann 2017). (4) The source information, i.e. the context words for the local model and the coherent entities for the global model, is insufficient or misleading and thus hurts the performance. For example, in a document about the FIS Freestyle Skiing World Cup, the mention 'World Cup' is resolved as 'FIS_Alpine_Ski_World_Cup' due to the lack of context and coherence information. In another document, the mention 'AMERICAN' is resolved as 'American_football' since many names of football teams appear around the mention and promote the rank of 'American_football' among other candidates, while the gold entity is 'United_States' instead.

Table 7: Accuracy on AIDA-B Splitted by Mention Prior of the Gold Entity in Cases where the Gold Entity Exists in the Candidate List

$P(e\|m)$ of Gold Entity	Mentions	Accuracy
≤ 0.01	368	57.34%
0.01 - 0.03	309	80.58%
0.03 - 0.10	278	75.18%
0.10 - 0.30	372	94.09%
> 0.30	3148	97.94%

8 Conclusion and Future Work

In this paper, we present a recurrent global model for NED. The solution is based on learning to rank with boosted tree model and manually engineered features, including novel features that identify highly distinctive terms that help improve disambiguation accuracy in an adaptive manner. Experiments on benchmark NED datasets suggest that our model outperforms or is comparable with the previous state-of-the-art work.

In our future work, we will investigate how to further improve the performance of the proposed method by incorpo-

rating additional features learned using some independent deep learning models (notably, the entity and type embeddings).

Acknowledgement This work is supported by ARC DPs 170103710 and 180103411, D2DCRC DC25002 and DC25003. The Titan V used for this research was donated by the NVIDIA Corporation.

References

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; and Schneider, N. 2013. Abstract meaning representation for sembanking. In *LAW@ACL*.

Barrena, A.; Soroa, A.; and Agirre, E. 2015. Combining mention context and hyperlinks from Wikipedia for named entity disambiguation. In **SEM*, 101–105.

Barrena, A.; Soroa, A.; and Agirre, E. 2016. Alleviating poor context with background knowledge for named entity disambiguation. In *ACL*.

Cao, Y.; Hou, L.; Li, J.-Z.; and Liu, Z. 2018. Neural collective entity linking. In *COLING*.

Cheng, X., and Roth, D. 2013. Relational inference for wikification. In *EMNLP*, 1787–1796.

Chisholm, A., and Hachey, B. 2015. Entity disambiguation with web links. *TACL* 3:145–156.

Dong, X. L.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; and Zhang, W. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, 601–610.

Francis-Landau, M.; Durrett, G.; and Klein, D. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *NAACL-HLT*, 1256–1261.

Gabrilovich, E.; Ringgaard, M.; and Subramanya, A. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Ganea, O.-E., and Hofmann, T. 2017. Deep joint entity disambiguation with local neural attention. In *EMNLP*, 2619–2629.

Ganea, O.-E.; Ganea, M.; Lucchi, A.; Eickhoff, C.; and Hofmann, T. 2016. Probabilistic bag-of-hyperlinks model for entity linking. In *WWW*, 927–938.

Globerson, A.; Lazic, N.; Chakrabarti, S.; Subramanya, A.; Ringgaard, M.; and Pereira, F. 2016. Collective entity resolution with multi-focal attention. In *ACL*, 621–631.

Guo, Z., and Barbosa, D. 2018. Robust named entity disambiguation with random walks. *Semantic Web* 9(4):459–479.

Gupta, N.; Singh, S.; and Roth, D. 2017. Entity linking via joint encoding of types, descriptions, and context. In *EMNLP*, 2681–2690.

Han, X., and Zhao, J. 1999. Nlpr kbp in tac 2009 kbp track: A two-stage method to entity linking. In *TAC*.

Han, X.; Sun, L.; and Zhao, J. 2011. Collective entity linking in web text: a graph-based method. In *SIGIR*, 765–774.

He, Z.; Liu, S.; Li, M.; Zhou, M.; Zhang, L.; and Wang, H. 2013. Learning entity representation for entity disambiguation. In *ACL*, 30–34.

Hoffart, J.; Yosef, M. A.; Bordino, I.; Fürstenau, H.; Pinkal, M.; Spaniol, M.; Taneva, B.; Thater, S.; and Weikum, G. 2011. Robust disambiguation of named entities in text. In *EMNLP*, 782–792.

Huang, H.; Heck, L. P.; and Ji, H. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *CoRR* abs/1504.07678.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A convolutional neural network for modelling sentences. In *ACL*, 655–665.

Kataria, S. S.; Kumar, K. S.; Rastogi, R. R.; Sen, P.; and Sengamedu, S. H. 2011. Entity disambiguation with hierarchical topic models. In *SIGKDD*, 1037–1045.

Kolitsas, N.; Ganea, O.; and Hofmann, T. 2018. End-to-end neural entity linking. *CoRR* abs/1808.07699.

Lazic, N.; Subramanya, A.; Ringgaard, M.; and Pereira, F. 2015. Plato: A selective context model for entity resolution. *TACL* 3:503–515.

Le, P., and Titov, I. 2018. Improving entity linking by modeling latent relations between mentions. In *ACL*, 1595–1604.

Ling, X.; Singh, S.; and Weld, D. S. 2015. Design challenges for entity linking. *TACL* 3:315–328.

Ma, C.; Chowdhury, F. A. R. R.; Deshwal, A.; Islam, M. R.; Doppa, J. R.; and Roth, D. 2019. Randomized greedy search for structured prediction: Amortized inference and learning. In *IJCAI*, 5130–5138.

Nguyen, D. B.; Theobald, M.; and Weikum, G. 2016. J-NERD: Joint named entity recognition and disambiguation with rich linguistic features. *TACL* 4:215–229.

Pan, X.; Cassidy, T.; Hermjakob, U.; Ji, H.; and Knight, K. 2015. Unsupervised entity linking with abstract meaning representation. In *NAACL-HLT*.

Ratinov, L.; Roth, D.; Downey, D.; and Anderson, M. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL-HLT*, 1375–1384.

Shen, W.; Wang, J.; and Han, J. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *TKDE* 27:443–460.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2008. Yago: A large ontology from wikipedia and wordnet. *J. Web Semant.* 6:203–217.

Sun, Y.; Lin, L.; Tang, D.; Yang, N.; Ji, Z.; and Wang, X. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *IJCAI*, 1333–1339.

Xue, M.; Cai, W.; Su, J.; Song, L.; Ge, Y.; Liu, Y.; and Wang, B. 2019. Neural collective entity linking based on recurrent random walk network learning. In *IJCAI*, 5327–5333.

Yamada, I.; Shindo, H.; Takeda, H.; and Takefuji, Y. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *SIGNLL*, 250–259.