A Hybrid Neural Model for Type Classification of Entity Mentions

Li Dong[†*] **Furu Wei**[‡] **Hong Sun**[$] **Ming Zhou**[‡] **Ke Xu**[†]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[‡]Microsoft Research, Beijing, China
[$]Microsoft Corporation, Beijing, China
donglixp@gmail.com {fuwei,hosu,mingzhou}@microsoft.com kexu@nlsde.buaa.edu.cn

Abstract

The semantic class (i.e., type) of an entity plays a vital role in many natural language processing tasks, such as question answering. However, most of existing type classification systems extensively rely on hand-crafted features. This paper introduces a hybrid neural model which classifies entity mentions to a wide-coverage set of 22 types derived from DBpedia. It consists of two parts. The mention model uses recurrent neural networks to recursively obtain the vector representation of an entity mention from the words it contains. The context model, on the other hand, employs multilayer perceptrons to obtain the hidden representation for contextual information of a mention. Representations obtained by the two parts are used together to predict the type distribution. Using automatically generated data, these two parts are jointly learned. Experimental studies illustrate that the proposed approach outperforms baseline methods. Moreover, when type information provided by our method is used in a question answering system, we observe a 14.7% relative improvement for the top-1 accuracy of answers.

1 Introduction

The type of an entity is very useful for various natural language processing tasks, such as question answering [Murdock *et al.*, 2012], and relation extraction [Ling and Weld, 2012]. The task of type classification aims to classify an entity mention in a specific context to a wide-coverage set of types. This task is non-trivial. First, entity mentions with surface names are highly ambiguous. For instance, the mention text "*Gates*" appears in the sentences "[*The greater part of*][*Gates*][*' population is in Marion County.*]" and "[*Gates*][*was a baseball player.*]". We need to classify the first mention to *Location*, and the other one to *Person*. Second, the compositional nature of entity mentions bring both challenges and opportunities to the type classification task. For example, the mention "*Bill & Melinda Gates Foundation*" belong to *Organization*. However, most of the words ("*Bill*", "*Melinda*", "*Gates*") indicate that its type is *Person*, which misleads bag-of-words methods. If the compositionality is considered, the composition of a person name phrase and "*Foundation*" can be correctly classified to the *Organization* class even if it is uncommon or absent in training data.

The mainstream methods [Rahman and Ng, 2010; Yosef *et al.*, 2012] model this problem as a classification task. Different classifiers (such as SVM, and MaxEnt) with extensive feature engineering are employed. These approaches heavily rely on hand-crafted features and external resources, e.g., POS tags, dependency relations, gazetteers. We address this by introducing a neural model to automatically obtain representations of a mention and its context. The model learns to embed the supervisions into word vectors, and builds representations from words to phrases. In addition, these bag-of-words methods do not utilize the compositional nature of language as the above examples. It limits their abilities to generalize for uncommon or unseen mentions. Our model learns a global composition matrix to recursively perform semantic compositions for entity mentions. It enables the model to learn some composition patterns for the type classification.

Specifically, we introduce a neural model to predict types for entity mentions. The model is based on the automatically learned distributed representations of mentions and contexts. The mention model is built upon recurrent neural networks. It recursively performs semantic compositions to obtain vector representations of mentions from word vectors. The context model utilizes multilayer perceptrons to compute hidden representations of contextual information. Next, their representations are jointly used to predict the type distribution. In addition, we use the DBpedia ontology to derive a wide-coverage set of types. Wikipedia anchor texts are utilized to automatically generate training data, which avoids expensive hand-annotation efforts. Extensive experiments are conducted on the automatically generated data and manually annotated data to compare with baseline methods and previous systems. The experimental results illustrate that our method outperforms baselines. Compared with previous work, our method yields better results without using feature engineering and external resources. We also integrate our method into a question answering system, and there is a 14.7% relative improvement for the top-1 accuracy.

The major contributions are three-fold:

- We introduce a hybrid neural model for the type classifi-

[*]Contribution during internship at Microsoft Research.

cation to automatically learn representations of mentions and context words without using hand-crafted features;

- We provide a new way to utilize the compositional nature of entity mentions for this task, which enables the model to better generalize for uncommon or unseen mentions;
- We present empirical studies on both type classification task and question answering task to evaluate the effectiveness of our method.

2 Related Work

Most of state-of-the-art tools for Named Entity Recognition (NER) only support a small set of types, such as *Location*, *Person*, *Organization*, and *Misc*. However, sometimes it is not enough for end-to-end tasks. In the question answering task, questions are classified into much more answer types [Li and Roth, 2002]. Answers are extracted and ranked using a rich set of features. The type matching score between a question and its answer candidates is one of the most important features. In other words, we need to know whether the candidate answers belong to *Event*, *Food*, or *Vehicle* instead of *Misc* which is too general. Another widely used approach decomposes the typing problem into two stages. Firstly, a Named Entity Linking (NEL) tool is used to link natural language phrases to entities of a knowledge base. Then, their types are obtained by querying the knowledge resources. However, the performance drops for uncommon entities [Ling and Weld, 2012] because this method only works for the entity mentions which appear in the used knowledge base. For instance, in a question answering system, the answer extraction algorithm does not guarantee that extracted answer candidates appear within a knowledge base. Besides, the NEL is a harder problem than predicting mention types, so its computation costs are higher for acceptable accuracy.

There has been some existing research focused on classifying natural language mentions into a richer set of lexical types. Fleischman and Hovy [2002] utilize a decision tree classifier to classify mentions into eight subtypes of *Person* class. It uses contextual word features and WordNet synonyms to improve the coverage. Rahman and Ng [2010] propose to use collective classification to consider relations of entities in a given document. And it employs a rich set of features, such as morphological features, grammatical features, gazetteer-based features, and WordNet sense features. Ling and Weld [2012] use a conditional random fields model to jointly tag boundaries of entities and map their types to Freebase tags. It further uses the patterns obtained from the ReVerb system [Fader *et al.*, 2011] as features. Yosef *et al.* [2012] perform hierarchical classification using support vector machines, and classify mentions to the type taxonomy borrowed from the YAGO knowledge base. It also employs unigrams, bigrams, and trigrams appeared in the mention paragraph as additional topical clues. Moreover, Lin *et al.* [2012] and Nakashole *et al.* [2013] work on discovering and typing emerging entities from news streams or social media. Most of these approaches rely on hand-crafted features and external resources, such as part-of-speech tags, dependency parsing results, WordNet, patterns from ReVerb system, and gazetteers. Our work employs recurrent neural networks and multilayer perceptrons to learn distributed representations of words from data automatically. Furthermore, we take semantic compositions of mentions and word orders into consideration instead of using bag-of-words features.

The internal structure and compositionality of names have been used for cross-document coreference [Li *et al.*, 2004] and named entity clustering [Elsner *et al.*, 2009]. Charniak [2001] employs a Markov chain to learn different parts of people's names from coreference data. Elsner *et al.* [2009] build an unsupervised generative model for named entity clustering. This model aims at modeling the entity mention internal structure and clustering related words by role. These methods learn different components of names in a symbolic way. By contrast, our method addresses this problem by using a recurrent neural model. It learns a composition matrix to model the compositionality of names. The similar names are closer in the learned task-specific vector space. Moreover, these vector representations can be directly used as features to classify mentions to their types.

Recently, the deep learning has achieved some promising results for many NLP tasks [Collobert *et al.*, 2011; Chen and Manning, 2014; Dong *et al.*, 2014a; 2014b; 2015]. In this paper, we utilize recurrent neural networks [Elman, 1990; Mikolov *et al.*, 2010] to obtain vector representations for entity mentions, and multilayer perceptrons to model the context. Recurrent neural networks are effective for many NLP tasks as they better utilize the compositional nature of language. So it is intuitive to use recurrent neural networks to address the problem of linguistic creativity. Collobert *et al.* [2011] develop a neural model for the named entity recognition task. However, it does not take the compositionality of entity mentions into consideration, and only uses four entity tags (*Location*, *Person*, *Organization*, and *Misc*) instead of a richer taxonomy.

3 Hybrid Neural Model

To begin with, we state the type classification problem as follows. Given an entity mention and the words in a contextual window, our task is to predict its type. Formally speaking, the input is $[c_{-S} \dots c_{-1}][w_1 \dots w_n][c_1 \dots c_S]$, where S is the window size, c_i represents a context word, n is the length of mention, and w_i is a mention word. The *L* and *R* paddings are used for left and right absent context words respectively. We need to compute the distribution $\mathbf{y} \in \mathbb{R}^{C \times 1}$ for the C types, and the type with largest probability is regarded as the predicted label.

3.1 Overview

As shown in Figure 1, our approach consists of two parts, namely, the mention model and the context model. The mention model employs Recurrent Neural Networks (RNNs) to obtain vector representations for entity mentions. Given a composition order, RNNs recursively perform semantic compositions over the words of an entity mention. The vector of a phrase is recursively computed by the vectors of words in a bottom-up way. Then, the representation is used as features for the entity mention. The second part is the context

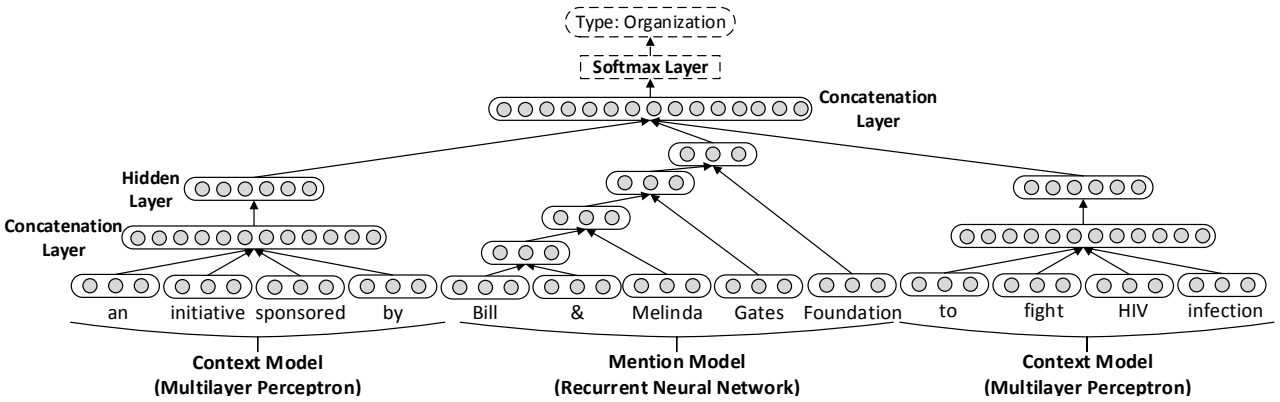

Figure 1: The prediction process for "[*an initiative sponsored by*][*Bill & Melinda Gates Foundation*][*to fight HIV infection*]". The neural network architecture consists of two parts. The mention model employs recursive neural networks to recursively obtain the vector representation for mention string. Moreover, the context model uses multilayer perceptrons to obtain hidden representations of context words.

model. MultiLayer Perceptrons (MLPs) are employed to utilize contextual information of a mention. Specifically, the words in a predefined contextual window are represented as vectors. Next, the concatenation vector of the word vectors goes through a hidden layer. Similarly, this vector obtained by the hidden layer is used as the representation of contextual information. Notably, word vectors are different in the mention model and the context model. In other words, they are regarded as different parameters, and are updated in the training process.

The learned representations of entity mention and its context are used as features. As shown in Figure 1, they are concatenated and fed into a softmax classifier to predict the type of entity mention. Specifically, softmax($\mathbf{z}$) outputs the probability distribution over C types. The h-th element of softmax($\mathbf{z}$) is $\frac{\exp\{\mathbf{z}_h\}}{\sum_j \exp\{\mathbf{z}_j\}}$. For the mention instance i, its predicted distribution is calculated via $\mathbf{y}^i = \text{softmax}\left(U\mathbf{x}^i\right)$ where U is the parameter matrix for classification, $\mathbf{x}^i$ is the concatenated vector representation, and $\mathbf{y}^i$ is the predicted distribution. The whole model is jointly trained, and its two parts are described as follows.

3.2 RNN-based Mention Model

Recurrent Neural Networks (RNNs), also called Elman networks [Elman, 1990], use D-dimensional vectors to represent words and phrases. They learn a global composition function and word embeddings from data. In order to compute vector representations for phrases, this composition function is recursively used to perform compositions in a given order. We define the composition order as from left to right. For instance, the representation of phrase "$w_1 w_2$" is computed via:

$$\mathbf{p} = f\left(W\begin{bmatrix}\mathbf{w}_1\\ \mathbf{w}_2\end{bmatrix} + \mathbf{b}^m\right) \tag{1}$$

where $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^{D\times 1}$ are D-dimensional word vectors, $W \in \mathbb{R}^{D\times 2D}$ is the composition matrix, $\mathbf{b}^m$ is the bias vector, and f is the nonlinearity function (such as tanh, sigmoid).

Equation 1 is recursively used to calculate vectors for mention phrases from left to right. As illustrated in Figure 1, the representation of "*Bill & Melinda Gates*" is calculated by the composition of "*Bill & Melinda*" and "*Gates*", and the representation of the whole mention "*Bill & Melinda Gates Foundation*" is recursively obtained by the vectors of "*Bill & Melinda Gates*" and "*Foundation*".

3.3 MLP-based Context Model

We use MultiLayer Perceptron (MLP) with one hidden layer to capture contextual information of an entity mention in the type classification task. The tokens in a contextual window are regarded as the context of a mention. The context words on the right side $c_1 c_2 \dots c_S$ are used to describe the model, and it is the same for the left side. Specifically, context words are represented by low-dimension vectors which are different from the ones in mention model. Firstly, these word vectors are concatenated. Then, it is fed into a hidden layer which produces a L-dimensional vector. The output of the hidden layer is computed via:

$$\mathbf{h} = f\left(H[\mathbf{c_1}^\mathsf{T} \dots \mathbf{c_S}^\mathsf{T}]^\mathsf{T} + \mathbf{b}^c\right) \tag{2}$$

where $\mathbf{c_1} \dots \mathbf{c_S} \in \mathbb{R}^{D\times 1}$ are D-dimensional word vectors, $H \in \mathbb{R}^{L\times DS}$ is the weight matrix, $\mathbf{b}^c$ is the bias vector, and f is the nonlinearity function.

To predict the type, the hidden representations of context words are used together with the vector of entity mention. In addition, they are jointly trained on data.

3.4 Model Training

The softmax classifier is employed to compute probabilities for C types. And the predicted distributions $\mathbf{y}^i$ are compared with ground truth $\mathbf{t}^i$ for instance i, where $\mathbf{y}^i, \mathbf{t}^i \in \mathbb{R}^{C\times 1}$. $\mathbf{t}^i_k$ is set to 1 if the correct type is k, and the others are 0. We minimize the regularized cross-entropy error between these two distributions. The objective function is:

$$\underset{\theta}{\text{minimize}} -\sum_i \sum_j \mathbf{t}^i_j \log \mathbf{y}^i_j + \frac{\lambda_\theta}{2}\|\theta\|_2^2 \tag{3}$$

Organisation, MeanOfTransportation, Holiday, Work, Food, Award, AnatomicalStructure, Device, Colour, Language, TopicalConcept, EthnicGroup, Currency, Disease, Drug, Person, Place, Activity, CelestialBody, Event, Species, BioChemSubstance

Table 1: Types derived from the ontology of DBpedia.

where λ_θ is the regularization parameter. The back-propagation algorithm [Rumelhart *et al.*, 1986] is used to jointly estimate parameters. It back-propagates errors of softmax classifier to other layers. Derivatives are calculated and gathered to train the model. The mini-batched AdaGrad [Duchi *et al.*, 2011] algorithm is then employed to solve this non-convex optimization problem.

3.5 Automatically Generating Training Data

We utilize DBpedia and anchor links in Wikipedia to automatically generate training data, which avoids expensive hand-annotation efforts. Similar idea was also used in [Nothman *et al.*, 2012; Ling and Weld, 2012]. Specifically, for a linked entity mention in Wikipedia, the mention string and context words are extracted. The anchor link of the entity mention helps us find its corresponding entity. Then, the type of this entity is queried from DBpedia. The entities which are not in DBpedia are ignored. We do not use Wikipedia's category information because these open categories are more like tags instead of well-defined types. The top-level categories of DBpedia ontology are employed in this paper. To make the types more specific, the type *Agent* is further expended to its subtypes (*Deity*, *Employer*, *Family*, *Organisation*, *Person*). As shown in Table 1, we obtain 22 top-level classes. Notably, the top-level types are disjoint. For instance, an entity is a writer and a singer, but its top-level type is still *Person*. So, if an entity has more than one top-level types that are automatically inferred by DBpedia, we use the most confident one as the type of this entity with the help of confidence ranking information provided by the DBpedia's type inference results.

4 Experiments

4.1 Datasets

To compare our method with baseline methods and previous work, we describe three datasets as follows.

Wiki-22: The 2014-03-04 Wikipedia dump and DBpedia 3.9 are used to generate the data. To compare with the previous methods, two million mentions are randomly sampled for training. Moreover, we use 0.1 million mentions as the dev set, and 0.28 million mentions as the test set.

Wiki-5: This dataset is introduced to evaluate the method HYENA in [Yosef *et al.*, 2012]. It is also automatically generated by using the Wikipedia and YAGO2 types.

News: This dataset is introduced to evaluate the method FIGER in [Ling and Weld, 2012]. It is manually annotated on 18 news reports.

4.2 Experiment Settings

The dev set is used to select hyper-parameters for our method and baselines. The nonlinearity function $f = \tanh$ is employed. The dimension of word vectors is set as 50. They are initialized by the pre-trained word embeddings provided by Turian *et al.* [2010]. The dimension of hidden layer of context model is 288. The parameters are initialized by the techniques described by Bengio [2012]. To train RNNs in the mention model, the gradients scaling down trick [Pascanu *et al.*, 2013] is used. For the context model, the size of context window is set as 6, i.e., there are at most 12 context words are considered. The regularization parameter λ_θ is set as 0.001. The learning rate used in AdaGrad is set as 0.01, and the mini-batch size is 10.

4.3 Evaluation Results

The micro-F1 score and macro-F1 score are used in this section to evaluate performances. For end-to-end applications, the macro-F1 score is more important than micro-F1 score. The micro/macro precision (P) and recall (R) are computed via:

$$P_{micro} = \frac{\sum_{i=1}^{C} |T_i \cap \hat{T}_i|}{\sum_{i=1}^{C} |\hat{T}_i|} \; R_{micro} = \frac{\sum_{i=1}^{C} |T_i \cap \hat{T}_i|}{\sum_{i=1}^{C} |T_i|} \quad (4)$$

$$P_{macro} = \frac{1}{C} \sum_{i=1}^{C} \frac{|T_i \cap \hat{T}_i|}{|\hat{T}_i|} \; R_{macro} = \frac{1}{C} \sum_{i=1}^{C} \frac{|T_i \cap \hat{T}_i|}{|T_i|} \quad (5)$$

where T_i is the set of mentions which belong to type i, and $\hat{T}_i$ is the set of mentions which are predicted to type i.

Comparison with Baseline Methods

Firstly, we compare our method with baseline methods on the test set of Wiki-22.

SVM. Support Vector Machine (SVM) is used in previous systems [Yosef *et al.*, 2012]. For the mention phrase and context words, unigram, bigram, and trigram features are employed. The LIBLINEAR [Fan *et al.*, 2008] tools are used.

MNB. Multinomial Naïve Bayes (MNB) is also a strong baseline for many tasks. The features are the same as in SVM, and Laplace smoothing is used.

ADD. It sums word embeddings to compute representations for mention model and context model.

HNM. The proposed Hybrid Neural Model in this paper.

We evaluate the models which only use mention features or context features. As shown in Table 2, mention features play more important roles than context features. Because mention phrases provide more explicit clues than contextual information for the type classification task. Moreover, the results demonstrate that our mention model and context model performs better than baselines. The HNM-mention employs RNNs to recursively obtain representations of entity mentions. It embeds the type information into word vectors and considers the semantic compositionality of mentions, which is better at classifying types. For the HNM-context, it learns a hidden representation from vectors of context words, and takes the word order into consideration. The results show that our method outperforms bag-of-words approaches. After jointly considering mention and context, our hybrid neural model achieves much better results than baselines.

Method	Micro-F1	Macro-F1
SVM-mention	90.2	89.7
MNB-mention	87.0	87.6
ADD-mention	90.1	90.7
HNM-mention	**93.4**	**93.6**
SVM-context	76.3	73.3
MNB-context	72.8	70.0
ADD-context	75.4	73.1
HNM-context	**81.1**	**78.3**
SVM-joint	93.5	93.4
MNB-joint	85.9	82.8
ADD-joint	94.1	93.9
HNM-joint (our)	**96.8**	**96.5**

Table 2: Evaluation results on dataset Wiki-22. -mention: Only mention feature template or mention model is used. -context: Only context feature template or context model is used. -joint: Both mention model and context model are used.

Dataset	Method	Micro-F1	Macro-F1
Wiki-5	HYENA	**95.2**	91.9
	HNM-joint	95.0	**93.6**
News	FIGER	72.6	80.1
	HNM-joint	**75.1**	**80.6**

Table 3: Evaluation results on the Wiki-5 and News datasets. Our method (HNM-joint) achieves comparable or better results than the previous systems HYENA and FIGER.

Comparison with Previous Systems

We also compare with the previous systems HYENA [Yosef *et al.*, 2012] and FIGER [Ling and Weld, 2012].

HYENA. This system uses unigrams, bigrams, and trigrams of mentions, surrounding sentences, and mention paragraphs as features. Moreover, part-of-speech tags of context words and gazetteer dictionary are also employed as features. SVM is used as the classifier.

FIGER. For entity mentions, unigrams, word shapes, part-of-speech tags, length, Brown clusters, head words, dependency structures are employed as features. They also use unigrams and bigrams of contextual sentences as features. Moreover, ReVerb patterns are employed. Perceptron is used as the classifier.

HYENA and FIGER provide their test datasets and predicted results. Consequently, we directly evaluate on their test data and use their provided predicted labels to compute evaluation metrics rather than re-implementing these two methods. The test datasets have been introduced as Wiki-5 and News in Section 4.1. In order to conduct a fair evaluation, the training data size used for our method is the same as theirs, and the test data are not included in the train split. Because the DBpedia ontology is used for our method, and the YAGO ontology is employed for HYENA. In order to compare results on HYENA's test data, a type mapping is manually performed to transform our 22 predicted types to the five top-level types (*Artifact, Event, Organization, Person, GeoEntity*) of HYENA. Similarly, for the comparison with FIGER, we map our types to the eight top-level types (*Organization, Art, Event, Person, Location, Product, Building, Others*) of FIGER. As described in Section 3.5, the top-level types should be disjoint, so we compute the evaluation metrics on the entity mentions assigned with one top-level type.

Method	Micro-F1	Macro-F1
SVM-mention	75.8	68.8
MNB-mention	75.5	69.0
ADD-mention	76.1	69.3
HNM-mention	**82.5**	**75.6**

Table 4: Evaluation results on long and unseen mentions in the Wiki-22 test set. Our RNN-based mention model outperforms baselines because it utilizes the compositional nature of mentions.

As shown in Table 3, our method achieves comparable or better performances than HYENA and FIGER without using hand-crafted features and external resources. Compared with HYENA on Wiki-5, the micro-F1 score of HNM-joint is comparable with HYENA, and the macro-F1 score of our method rises by 1.7%. Compared with FIGER on the News dataset, the micro-F1 score and macro-F1 score of HNM-joint increase by 2.5% and 0.5% respectively. The evaluation results indicate the effectiveness of our method.

Evaluation on Unseen Mentions

In order to illustrate the generalization ability of RNN-based mention model, we evaluate on the test mention phrases which do not appear in the train set and their lengths are greater than two. For these 20,224 unseen mentions, we compare our method to SVM, MNB, and ADD. As shown in Table 4, our RNN-based mention model achieves improvements than baselines. The results indicate that utilizing the compositionality helps us to deal with uncommon or unseen mentions. The improvements are larger than the results evaluated on all the test data.

4.4 Examples: Compositionality of Mentions

In order to demonstrate the compositional nature of mentions, we query some similar composition examples for the mentions in Wiki-22 test set. The cosine similarity is used as our similarity metric.

As shown in Table 5, the first case belongs to *Event*, and we find that its nearest compositions follow the same composition pattern. The second example is *Organization*, and all the results consist of a location name, "*University*", and "*School/College of Law*". We find that the mention model learns similar word representations for "*School*" and "*College*". The third case and its similar compositions are combinations of a person name and "*Award*". The next example belongs to *Disease*. The pattern of these mentions is the name of an organ followed by the name of a specific disease. The last mentions all belong to *Species*, and are in a same form. We notice that the mentions that are of similar patterns are closer. This indicates that RNNs learn how to recursively conduct compositions according to supervisions of type information. Compositions help the model generalize to uncommon or unseen mentions.

English civil war	Spanish civil war / Greek civil war / Nigerian civil war / Angolan civil war
Columbia University School of Law	Northwestern University School of Law / West Virginia University College of Law / University of Iowa College of Law / Golden Gate University School of Law
Subdural Hematoma	Intracranial Haemorrhage / Cardiac Arrhythmia / Duodenal Ulcer / Arterial Thrombosis
Joseph Jefferson Award	Margaret A. Edwards Award / Marian Engel Award / Doug Wright Award / Timothy Findley Award
Red-bellied Lemur	Oriental White-eye / Red-legged Honeycreeper / Black-crowned White-eye / Snowy Egrets

Table 5: We query some similar composition examples for the mentions in Wiki-22 test set. The cosine similarity of mentions' vector representations is used as the similarity metric. The mentions which are of similar patterns are closer.

4.5 Evaluation of Type Classification in Question Answering

In this section, we evaluate the effectiveness of type classification results in a web based question answering (QA) system. We follow the typical design of the web based QA system as in [Cucerzan and Agichtein, 2005; Lin, 2007]. We send the input question to a commercial search engine [1]. Then answer candidates are generated from titles and snippets of search results using the method described in [Chu-Carroll and Fan, 2011]. Finally, a rich set of features (such as similarity features, redundancy features, and appearance count features) are used to rank these answer candidates. We use SVM-rank [Joachims, 2006] to learn the answer ranker in our implementation. The research and development of the question answering system is beyond the scope of this paper. We are particularly interested in the application of the type classification results in the answer ranking component of our QA system.

Specifically, we add a feature template into the answer ranking module. We build a question classifier [Huang *et al.*, 2008] to classify a question q into 18 broad classes as its answer type T_q. The types of answer candidates T_a are obtained by the type classification algorithm. The interaction of the answer type and candidate type (i.e., $T_q|T_a$) is employed as a binary feature. The ranking model automatically learns whether these two types are matched or not. For instance, the answer type of question "*who is the ceo of microsoft?*" is *Person*, and the types of its answer candidates "*Satya Nadella*" and "*Xbox*" are *Person* and *Device* respectively. Consequently, their features for ranking model are "*Person*|*Person*" and "*Person*|*Device*" respectively.

We use the recently released WebQuestions dataset [Berant *et al.*, 2013] in our experiments. It contains 3,778 training instances and 2,032 test instances. The questions are collected by querying the Google Suggest API. A breadth-first search beginning with *wh-* is conducted. Then, answers are annotated by the workers in Amazon Mechanical Turk. We do not use the traditional TREC QA datasets in our experiments because the answers of many questions in the dataset have not been correct now for temporal issues.

This QA system always returns a ranking list of answers, so we use the Acc@k ($k = 1, 3, 5$) as the evaluation criterion. The Acc@k is the fraction of questions which obtain correct answers in their top-k results. As shown in Table 6, using our method in the answer ranking model makes the performance of w/TYPE become better than the w/oTYPE. To be specific, the top-1 accuracy of w/TYPE rises by 4.3% (i.e., 14.7% relative improvement) comparing with w/oTYPE. The Acc@3 and Acc@5 also increase by 4.8% and 3.2% respectively. This indicates that our method helps to improve the QA task and proves the effectiveness of our approach. Moreover, our method can also be used to directly prune answer candidates before ranking, which is not the focus of this work.

Method	Acc@1	Acc@3	Acc@5
w/oTYPE	29.2	50.8	61.2
w/TYPE	**33.5**	**55.6**	**64.4**

Table 6: Evaluation results on the QA task. Type information obtained by our approach improves the accuracy. w/oTYPE: Without using type features in the answer ranking model. w/TYPE: Using type features in the answer ranking model.

5 Conclusion and Future Work

We introduce a neural model to classify entity mentions to their corresponding types in this paper. We learn the vector representations of an entity mention and its context with recurrent neural networks and multilayer perceptrons respectively. Then they are used to jointly predict the type distribution. Furthermore, the Wikipedia anchor links and the DBpedia ontology are utilized to automatically generate training data.We conduct extensive experiments to compare our method with the baseline methods MNB and SVM. Experimental results show that our model improves the baselines. We also compare our method with the previous work (HYENA and FIGER). The results indicate that our method outperforms these two methods without hand-crafted features and external resources. Moreover, by integrating the type predictions of our method into the answer ranking model of a question answering system, we observe a 14.7% relative gain for the top-1 accuracy. In the future, several interesting directions are worth exploring. First, we can support more subtypes to achieve a fine-grained type classification. For example, the person class can be further classified to doctor, president, etc. Second, the global information (e.g., topic) has a correlation with the type distribution. So we can learn the representations of global texts and utilize them in this framework. In addition, we can apply this method in the relation extraction task to improve its performance.

[1] The Microsoft Bing search engine is used to retrieve the top 20 search results for each question in our experiments.

Acknowledgments

This research was partly supported by NSFC (Grant No. 61421003) and the fund of the State Key Lab of Software Development Environment (Grant No. SKLSDE-2015ZX-05).

References

[Bengio, 2012] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. 2012.

[Berant *et al.*, 2013] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP '13*, 2013.

[Charniak, 2001] Eugene Charniak. Unsupervised learning of name structure from coreference data. In *NAACL*, 2001.

[Chen and Manning, 2014] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, 2014.

[Chu-Carroll and Fan, 2011] Jennifer Chu-Carroll and James Fan. Leveraging wikipedia characteristics for search and candidate generation in question answering. In *AAAI*, 2011.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12, 2011.

[Cucerzan and Agichtein, 2005] Silviu Cucerzan and Eugene Agichtein. Factoid question answering over unstructured and structured web content. In *TREC*, volume 72, page 90, 2005.

[Dong *et al.*, 2014a] Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL*, pages 49–54, 2014.

[Dong *et al.*, 2014b] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *AAAI*, 2014.

[Dong *et al.*, 2015] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over freebase with multi-column convolutional neural networks. In *ACL*, 2015.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[Elman, 1990] Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 1990.

[Elsner *et al.*, 2009] Micha Elsner, Eugene Charniak, and Mark Johnson. Structured generative models for unsupervised named-entity clustering. In *NAACL*, 2009.

[Fader *et al.*, 2011] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP '11*, July 27-31 2011.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *JMLR*, 9, June 2008.

[Fleischman and Hovy, 2002] Michael Fleischman and Eduard Hovy. Fine grained classification of named entities. In *COLING '02*, 2002.

[Huang *et al.*, 2008] Zhiheng Huang, Marcus Thint, and Zengchang Qin. Question classification using head words and their hypernyms. In *EMNLP*, 2008.

[Joachims, 2006] Thorsten Joachims. Training linear svms in linear time. In *SIGKDD*, 2006.

[Li and Roth, 2002] Xin Li and Dan Roth. Learning question classifiers. In *COLING*, pages 1–7, 2002.

[Li *et al.*, 2004] X. Li, P. Morie, and D. Roth. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *AAAI*, pages 419–424, 2004.

[Lin *et al.*, 2012] Thomas Lin, Oren Etzioni, et al. No noun phrase left behind: detecting and typing unlinkable entities. In *EMNLP-CoNLL*, 2012.

[Lin, 2007] Jimmy Lin. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Trans. Inf. Syst.*, 25(2), April 2007.

[Ling and Weld, 2012] X. Ling and D.S. Weld. Fine-grained entity recognition. In *AAAI*, 2012.

[Mikolov *et al.*, 2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

[Murdock *et al.*, 2012] J William Murdock, Aditya Kalyanpur, Chris Welty, James Fan, David A Ferrucci, DC Gondek, Lei Zhang, and Hiroshi Kanayama. Typing candidate answers using type coercion. *IBM Journal of Research and Development*, 56, 2012.

[Nakashole *et al.*, 2013] Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. Fine-grained semantic typing of emerging entities. In *ACL*, 2013.

[Nothman *et al.*, 2012] Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194, 2012.

[Pascanu *et al.*, 2013] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318, 2013.

[Rahman and Ng, 2010] Altaf Rahman and Vincent Ng. Inducing fine-grained semantic classes via hierarchical and collective classification. In *COLING '10*, 2010.

[Rumelhart *et al.*, 1986] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088), 1986.

[Turian *et al.*, 2010] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *ACL*, 2010.

[Yosef *et al.*, 2012] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. Hyena: Hierarchical type classification for entity names. In *COLING '12*, 2012.