IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 21, 2017, accepted July 15, 2017, date of publication July 25, 2017, date of current version August 22, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2731761

Entity Search Based on the Representation Learning Model With Different Embedding Strategies

SHIJIA E, (Student Member, IEEE), AND YANG XIANG
College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China

Corresponding author: Yang Xiang (shxiangyang@tongji.edu.cn)

This work was supported in part by the National Basic Research Program of China under Grant 2014CB340404, in part by the National Natural Science Foundation of China under Grant 71571136, and in part by the Project of Science and Technology Commission of Shanghai Municipality under Grant 16JC1403000 and Grant 14511108002.

ABSTRACT We focus on the problem of learning distributed representations for entity search queries, named entities, and their short descriptions. With our representation learning models, the entity search query, named entity, and description can be represented as low-dimensional vectors with minimal human preprocessing. Our goal is to develop a simple but effective model that can make the distributed representations of query-related entities similar to the query representation in the vector space. Hence, we propose three kinds of learning strategies, and the difference between them mainly lies in how to deal with the relationship between an entity and its description. We analyze the strengths and weaknesses of each learning strategy and validate our methods on public data sets, which contain various query types and different languages (i.e., English and Chinese). The experimental results indicate that our proposed methods can adapt to different types of entity search queries, and outperform the current state-of-the-art methods no matter the entity collection is homogeneous or heterogeneous. Besides, the proposed methods can be trained fast and can be easily extended to other similar tasks.

INDEX TERMS Entity search, entity embedding, language model, representation learning.

I. INTRODUCTION

In the research of the search engine, how to make the search system understand the user intentions behind the user queries is a crucial question [1]. An intelligent search system should meet either precise or vague requirement from users [2]. The returned searching results should be semantically relevant to the user queries not only with the simple word matching. In all kinds of query intentions, the search for entities is the most common search behavior. An entity search query that may be a keyword or a key phrase given by the users and the results returned by the system are sometimes composed of two parts. One part is named entity itself, and the other part is a brief description of the named entity. The entity search task is to automatically obtain at least one entity that matches the entity search query from all the available entities. The entity search based techniques can make the entities that semantically relevant to the entity search question rank at the forefront of the list of candidate entities. The entity search query, named entity and description can be referred to as the three elements involved in the process of entity search. The traditional methods based on rules need much work of feature engineering to obtain the semantic meaning of a word or a sentence. Due to the flexibility in short text, artificially defined rules cannot cover all features. Therefore, it needs much manual intervention to make the query results better and better.

Another problem in the entity search task is that for a generic entity search system, the user's input queries and candidate entities may contain different languages. Because of the different rules of grammar, we cannot use same dependency grammar to analyze phrase structure and semantic information. This limitation also makes the system need much human intervention.

Fortunately, the representation learning technology represented by deep learning is nice to settle the feature engineering problem. We can develop an end-to-end framework with the help of deep neural networks (DNNs). The DNNs try their best way to represent the meaning of a word or sentence, and then we can know the relationships among the words with the vector representations. In this paper, we use different embedding strategies to learn the distributed representations of entity search queries, named entities, and their descriptions. The probability distribution of observing a word depends on some fixed number of surrounding words with neural language models (NLM) [3]. However, we focus on using the named entities themselves and their descriptions to learn the implicit relationships between the candidate entities and the entity queries by NLM. It means our proposed model emphasizes that the entity embeddings should fit different query intentions. With the specific embeddings, the matching degree will express more semantic relationships than just using the vanilla word embeddings. Hence, we build an entity search framework to sort the searching results based on the learned semantic similarities. The performance of the proposed framework is better than the vanilla word embedding method [4]. Our work is most related to [5], and it learns the vector representations of questions and answers with convolutional neural networks (CNN), and it hopes the question vectors and their correct answer vectors could be close in the vector space. However, the length of query statements, and especially the named entities with their descriptions is short. Therefore, the CNN can not give full play to its role in the feature extraction. In our work, we trained the models to give the matching score between the entity search queries and the candidate entities. We optimized the models using adaptive moment estimation (Adam) [6] which is an enhanced variant of stochastic gradient descent.

As mentioned above, three basic elements are involved in the process of entity search, namely the entity search query, named entity, and entity description. We take the entity search query as a complete sentence, but for the named entity and entity description, we propose three different strategies to train the embeddings.

- Concat named entity and entity description together into a *sentence* as a complete candidate answer to the entity search query.
- Treat entity description and named entity together as an independent word (which means that this part will not be applied to word segmentation).
- Learn the embeddings of named entity and entity description respectively. We make the entity description correspond to a translation from named entity to entity search query.

The experimental results validate that the different embedding strategies can fit different kinds of entity queries. Our work makes some contribution to the research area of entity search using representation learning methods. First, we propose multiple embedding strategies which can learn better low-dimensional vector representations of words in the named entities, query statements and entity descriptions. Second, we validate that the dynamic word embeddings, *i.e.*, using a pre-trained word embeddings as the initial weights and making the word embeddings able to be updated dynamically during the training process can improve the performance of entity search task. The dynamic word embeddings imply the effective semantic relationships between entities and queries. Moreover, we present that our model is multi-language compatible, and it does not need to rely on language-related information. Finally, our proposed model can be extended to other similar tasks, and it is easy to add more layers to learn more complex features.

The rest of this paper is structured as follows. Section 2 contains related work; In Section 3, we formulate the problem and describe the model architectures used in this work. Experimental results and discussions are presented in Section 4, and finally, we give some concluding remarks in Section 5.

II. RELATED WORK

Our work is related to the neural network language model (NNLM), the answer selection methods of Q&A, and learning to rank for entity search. The core of them is using the distributed word representations. Recently, it also has been applied to several natural language processing (NLP) tasks, such as named entity recognition [7] and question answering [8]. Also, the idea of distributed word representations can be generalized to model sentences, paragraphs or even documents [9]. Lai *et al.* [10] compare various word embedding models on different tasks. It is a good guideline for training the word embeddings. From that work, we can know that the corpus domain is essential to generate meaningful word embeddings for a given task. Lee and Dernoncourt [11] and dos Santos and Gatti [12] propose a CNN based method with word embeddings to solve the short text classification task. Van Gysel *et al.* [13] introduce a latent vector space model to learn the distributed representations of words, products and a mapping between them. The model can learn excellent product representations to enhance the product search performance. As for the answer selection task, it is similar to the entity search problem, *i.e.*, given a question and an answer set for the question, the task is to find the best candidate answer(s). References [5], [14], and [15] design a few architectures of DNNs using CNN and Long Short-Term Memory networks (LSTM) to solve the problem. However, different from the answer selection task, the length of a named entity with its description is much shorter than a typical answer.

A. NNLM

A neural networks language model predicts the probability distribution of the next word utilizing several previous words [3]. For a training sample $(w_1, w_2, \ldots, w_k)$ in the corpus, the goal of the model is to maximize the log-likelihood of

$$p(w_k|w_1, w_2, \ldots, w_{k-1}) \tag{1}$$

where w_k (the k^{th} word in the input sequence) is the target word we need to predict. Figure 1 shows the basic structure of the NNLM. In this model, the previous words together are called the context to the word w_k, and the model concatenates the context's embeddings as the input. The output softmax layer consists of N units, where N denotes

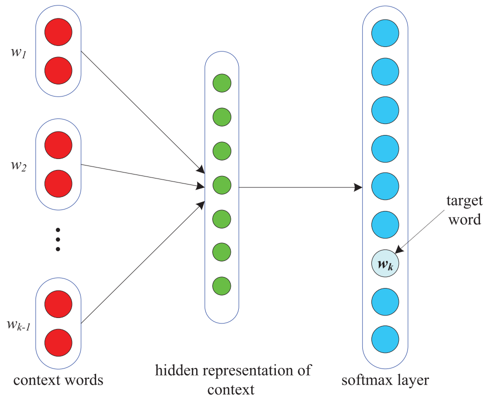


FIGURE 1. The overall model structure of the NNLM.

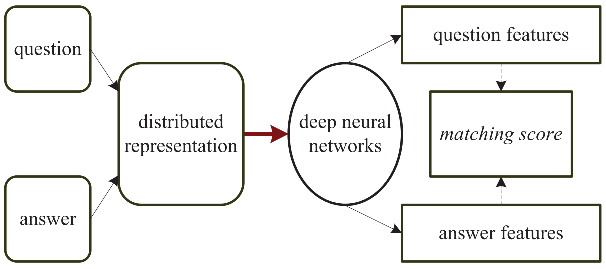


FIGURE 2. The solution framework of answer selection.

the vocabulary size of the corpus, and the model tries to predict the target word w_k with the highest probability. The challenge of the basic model is that the computational overhead is expensive. Hence, Morin and Bengio [16] and Mnih and Teh [17] use hierarchical softmax and noise contrastive estimation respectively to help reduce the training duration. Collobert and Weston [18] propose a model known as the C&W model where the central word in a sequence is the target word, and the surrounding words are put together into a context. Unlike the basic NNLM model, C&W model combines the context and target word and then give a score. Therefore, the training target is that the score of correct target word should be higher than a noise word's score. This method is similar to the answer selection task. In the following, we review the method used which give some experience that we can absorb into our work.

B. ANSWER SELECTION

As mentioned above, the goal of the answer selection task is to find the best candidate answer. If the selected answer is contained in the ground truth set of the corresponding question, the prediction result is considered to be correct. Otherwise, it is incorrect. For this reason, the task can be treated as a binary classification problem. In addition to the distributed representation of questions and answers, another important thing is to give a metric to measure the matching degree of the Q&A pairs. The general solution framework is shown in Figure 2.

Feng *et al.* [5] present a framework based on CNN. The questions and answers share the same CNN layers to represent the features. It also attempts several general similarities metrics such as cosine similarity. Similarly, [19]–[21] propose the models based on CNN for matching natural language sentences. Tan *et al.* [14] consider the shortcomings of CNN and adopt the LSTM to model the Q&A pairs. LSTM is essentially a recurrent neural network (RNN), and the learned features can retain the word order, so as to further improve the overall performance of the model. To capture the contextualized local information in the matching process, Wan *et al.* [22] also present a deep architecture to match two sentences with multiple positional sentence representation. Our model proposed in this paper is not same as the answer selection task. In fact, due to the short text characteristic of named entities or brief descriptions, the current text matching methods cannot be directly used in the entity search task.

C. LEARNING TO RANK FOR ENTITY SEARCH

Learning to rank approaches are expected to learn a similarity function between pairs of objects. Severyn and Moschitti [23] present a convolutional neural network architecture for reranking text pairs. The model greatly improves on the previous state-of-the-art system with minimal preprocessing. Graus *et al.* [24] propose a model with combining entity descriptions from various knowledge bases to construct dynamic entity representations. It improves retrieval effectiveness by 7% over a learning to rank baseline. Chen *et al.* [25] use two learning to rank models: *RankSVM*, which is a pairwise method, and *Coordinate Accent*, which is a listwise method to solve the entity search problem. They first use the fielded sequential dependency model (FSDM) which is the previous state-of-the-art method as the base retrieval model [26], and the proposed ranking methods are used to rerank the top 100 entities per query retrieved by FSDM. The experimental results show that the learning to rank model can significantly improve the FSDM method. Besides, Hasibi *et al.* [27] discuss that incorporating entity linking for entity retrieval and present a model based on random markov fields as an extension to various entity retrieval models. Their approach can achieve significant improvements over previous models. Therefore, the learning to rank models can be built as an add-on of traditional entity search methods, and optimize the results of previous entity search algorithms through the supervised learning.

III. ENTITY SEARCH MODEL

In this section, we present our representation learning model for entity search task. The design is inspired by the distributed word representations and answer selection with deep neural networks. We learn distributed representations for entity search queries, candidate entities and also their descriptions (if exist) in a low-dimensional vector space. Unlike the typical answer selection task presented by [5] in Q&A research area, we exploit various embedding strategies and consider the short text characteristics of entities and descriptions. Moreover, we regard the entity description as a bridge connecting the named entity and search query. It is worth noting that the proposed model can still have a good performance when the entity description is missing.

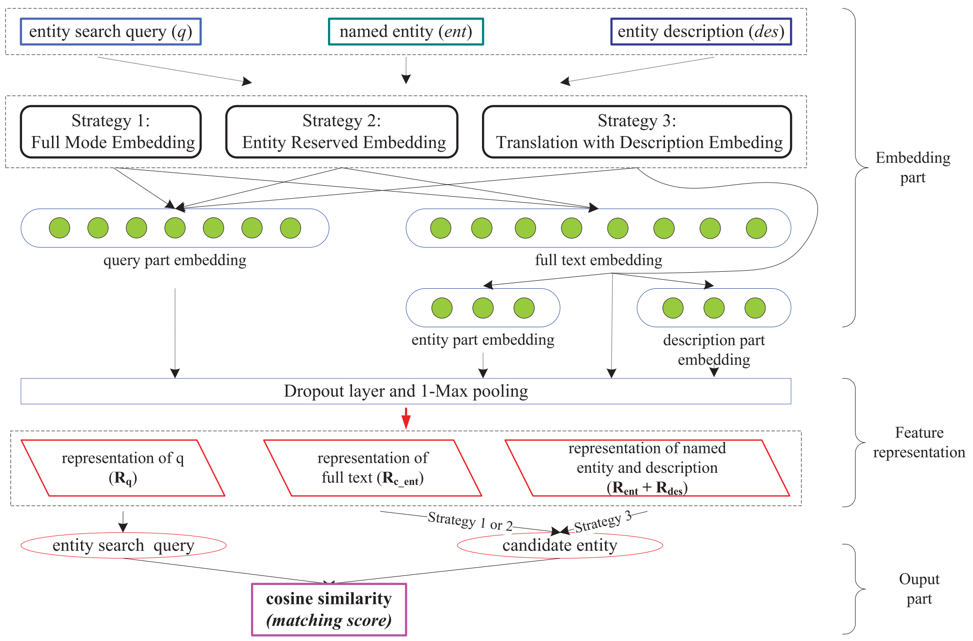

entity search query (q)
named entity (ent)
entity description (des)
Strategy 1: Full Mode Embedding
Strategy 2: Entity Reserved Embedding
Strategy 3: Translation with Description Embeding
Embedding part
query part embedding
full text embedding
entity part embedding
description part embedding
Dropout layer and 1-Max pooling
representation of q (R_q)
representation of full text (R_c_ent)
representation of named entity and description (R_ent + R_des)
Feature representation
Strategy 1 or 2
Strategy 3
entity search query
candidate entity
cosine similarity (matching score)
Ouput part

FIGURE 3. The overall architecture with different embedding strategies.

A. PROBLEM DEFINITION

To describe the model conveniently, we first define the concept involved in the entity search problem.

Definition 1 (Entity Search Query): The entity search query reflects the query intention. Let $q = (v_1, v_2, \dots, v_i)$ denote the entity search query, where v is a single word in the vocabulary list V. We can use $\boldsymbol{v}$ to denote the embeddings of v. i is the sequence length after word segmentation.

Definition 2 (Named Entity): A named entity is something that exists as itself. It can be an organization, a place or a person. Let $ent = (v_1, v_2, \dots, v_j)$ denote the word sequence of a named entity. j is the sequence length. The reason a named entity is considered as a sequence of words not just an independent entity is that a named entity may include words expressing some important information. For example, the name of a film such as *Pirates of the Caribbean* can reflect an individual style of the film itself.

Definition 3 (Entity Description): The entity description is a phrase about some features of the named entity, such as *Swimming athlete* or *Government official*. Let $des = (v_1, v_2, \dots, v_k)$ denote the entity description. It is just similar to the q, but not all ent have their corresponding descriptions. Therefore, the sequence length k maybe equal to zero.

Definition 4 (Candidate Entity): A candidate entity denoted as c_ent is composed of ent and des. When a named entity does not have its short description, ent itself is represented as c_ent.

Definition 5 (Candidate Pool Size): For a given q, the number of c_ent is called candidate pool size, denoted as p_size.

B. MODEL ARCHITECTURE

In this section, we demonstrate the three proposed embedding strategies. As shown in Figure 3, the embedding part aims to learn the basic word embeddings of v in q, ent, and des with different embedding strategies. The learned word embedding matrix can give each word a dense vector representation. The dropout layer is a regularization technique for reducing overfitting by preventing complex co-adaptations on training data [28]. With the 1-Max pooling layer, we get the distributed representation of each element. At last, by calculating the similarity between the representation of q and c_ent in the vector space, we obtain the matching score of them. Our representation learning method will do its best to represent the entity search problem while learning how to solve the problem, and the proposed embedding strategies describe the entity search problem from different perspectives. The following is a detailed description of the proposed model.

1) STRATEGY 1: FULL MODE EMBEDDING

The full mode embedding (FME) is the basic way to model q, ent, and des. With this strategy, ent and des are concatenated as a candidate entity (c_ent) to learn the corresponding representations. For each entity search query q, there is at least a golden standard candidate entity c_ent^+ regarded as a positive entity. A training instance is constructed by pairing this c_ent^+ with a negative entity c_ent^- sampled from all the candidate entities to the corresponding q. For each q, if there are multiple positive candidate entities, the c_ent^+ will not always be the same one to a negative entity.[1] As shown

[1]Briefly speaking, if one q has 100 candidate entities where five entities are positive (c_ent^+) and ninety-five entities are negative (c_ent^-), there will be 475 (95×5) training instances constructed with that q.

Algorithm 1 Translation With Description Embedding

```
Input: All $q$ in training set and their corresponding $(ent^+, des^+)$ and $(ent^-, des^-)$, margin $m$, batch size $b\_size$, learning rate $\alpha$,
    and embeddings size $n$.
Output: All the embeddings of $v \in V$ which can be used to express $q$, $ent$, and $des$.
 1: $\boldsymbol{v} \leftarrow$ uniform $(-0.05, 0.05)$ for each $v \in V$
 2: $\boldsymbol{v} \leftarrow \frac{\boldsymbol{v}}{|\boldsymbol{v}|}$
 3: repeat
 4:     $q_{batch} \leftarrow (q_1, q_2, \ldots, q_{b\_size})$ // sample a mini-batch of size $b\_size$
 5:     $ent^+_{batch} \leftarrow (ent^+_1, ent^+_2, \ldots, ent^+_{b\_size})$
 6:     $des^+_{batch} \leftarrow (des^+_1, des^+_2, \ldots, des^+_{b\_size})$
 7:     $Train_{batch} \leftarrow \emptyset$
 8:     for all $q \in q_{batch}$ do
 9:         Negative sampling: $ent^-_{batch} \leftarrow (ent^-_1, ent^-_2, \ldots, ent^-_{b\_size})$
10:         Negative sampling: $des^-_{batch} \leftarrow (des^-_1, des^-_2, \ldots, des^-_{b\_size})$
11:         $Train_{batch} \leftarrow Train_{batch} \cup \{[(q, ent^+_1, des^+_1), (q, ent^-_1, des^-_1)], \ldots, [(q, ent^+_{b\_size}, des^+_{b\_size}), (q, ent^-_{b\_size}, des^-_{b\_size})]\}$
12:     end for
13:     $R_q$, $R_{ent}$ and $R_{des}$ $\leftarrow$ Representation learning with Strategy 3 // Also can be Strategy 1 or 2 for FME and TDE.
14:     Update representation parameters w.r.t
            $\sum_{((q,ent^+,des^+),(q,ent^-,des^-))\in Train_{batch}} \nabla max\{0, m - cos(R_{ent^+} + R_{des^+}, R_q) + cos(R_{ent^-} + R_{des^-}, R_q)\}$
15: until convergence
```

in Figure 3, the model will generate the representations for q, c_ent^+, and c_ent^- denoted as R_q, $R_{c\_ent^+}$ and $R_{c\_ent^-}$. As a result, we minimize a ranking objective function defined as follows:

$$L = max\{0, m - cos(R_q, R_{c\_ent^+}) + cos(R_q, R_{c\_ent^-})\} \quad (2)$$

where m is a positive margin. Our goal is to make the positive entity more closed to the search query (higher similarity) than any negative entity in the vector space. If $cos(R_q, R_{c\_ent^-}) - cos(R_q, R_{c\_ent^+}) >= 0$, it means the $R_{c\_ent^-}$ will not be ranked below the $R_{c\_ent^+}$, so $m - cos(R_q, R_{c\_ent^+}) + cos(R_q, R_{c\_ent^-}) > 0$, and the neural network needs to update the parameters and a new negative example is sampled randomly. If $cos(R_q, R_{c\_ent^-}) - cos(R_q, R_{c\_ent^+}) < 0$, it means that the training process makes the positive sample is more suitable for the entity search question than the negative sample. However, only if it still be less than 0 after plus m, the loss L will be 0, and that is the best training result. In other words, the margin m is a hyper-parameter to control the distinguishability of the positive and negative entities, *i.e.*, we hope our system can distinguish between positive and negative entities as much as possible.

2) STRATEGY 2: ENTITY RESERVED EMBEDDING

The strategy 2, entity reserved embedding (ERE) pays more attention to the named entity itself, so we do not split *ent* which exists in *c_ent*. However, *des* can still be segmented into several v. This strategy is mainly to investigate whether a single named entity is sufficient to contain valid semantic information. As Figure 3 shown, the other parts of ERE are just same as FME.

3) STRATEGY 3: TRANSLATION WITH DESCRIPTION EMBEDDING

There are two parts in the first two strategies, *i.e.*, the query part q and the answer part composed of *ent* and *des*. Translation with description embedding (TDE) shown in Figure 3 is different from previous strategies. q, *ent*, and *des* are separately mapped to points in the vector space, and TDE tries to accurately describe the relationship among the three elements by vector operation. More formally, the model expects R_{des}, the distributed representation of *des* to be a translation from R_{ent} to R_q. Under this assumption, $R_{ent^+} + R_{des^+}$ should be the closest point of R_q, while R_q should be away from $R_{ent^-} + R_{des^-}$. To learn such representations, the objective function is:

$$L = max\{0, m - cos(R_{ent^+} + R_{des^+}, R_q) + cos(R_{ent^-} + R_{des^-}, R_q)\} \quad (3)$$

The detailed optimization procedure of TDE is described in Algorithm 1. At each iteration, a small set of training samples serves as the mini-batch. FME and ERE also follow this mini-batch training for stochastic optimization. More detailed information about the parameter settings will be presented in Section 4.

4) FEATURE REPRESENTATION AND SIMILARITY METRIC

With different embedding strategies, q, *ent*, and *des* have their distributed representations. Then, the dropout layer is to improve the generalization ability of the model. q, *ent*, and *des* may contain more than one v, so we use 1-Max pooling to select the maximum one of the embeddings as the final representation. For the entity search queries and candidate entities,

TABLE 1. Statistics of the four types of entity collections in Chinese dataset.

Datasets	tvShow	movie	restaurant	celebrity
# Entities (e)	9254	24347	54539	52831
# Entity search queries (q) (Train/Test)	100/1000 for each type			
# Candidate entities per q	∼100			
# Vocabulary	9840	19671	48158	52735
Out of vocabulary rate (unseen entities rate)	0	19.20%	11.70%	0.02%

the text length is really short, and the semantic information is mainly contained in the core word. Especially, the entity may consist of a single word. Therefore, the 1-Max pooling can effectively obtain the information. In our experiments, other pooling strategies such as the average pooling do not achieve better results than 1-Max pooling.

What we do in the last step is calculate the cosine similarity between the two representation vectors. In our experiments, we also tried other similarity metrics such as manhattan distance or euclidean distance, but the results were not better than the cosine similarity. Another important reason we use cosine similarity is that the inner product facilitates the derivation of gradients [29].

IV. EXPERIMENTS

In this section, we describe various experiments to evaluate the proposed representation learning model with different strategies. The Chinese datasets[2] and English dataset[3] are publicly available.

A. DATASETS

The Chinese dataset used in our experiments comes from Baidu Cup' 16. It consists of four types of entity collections (as shown in Table 1).

tvShow: In this dataset, e is a keyword or key phrase which belongs to some TV show related attributes, *e.g.*, q can be "*religious subjects*," and its candidate entities ($p_{size} \sim 100$) are selected from the TV show entity set. All of the e have their d, which describe the year that the TV show is on.

Movie: The *movie* dataset is same as the *tvShow* dataset, and only the named entity is changed into a movie. All named entities also have the description about the time of the movie's release.

Restaurant: This dataset is different from the previous two. The q in the dataset is about some characteristics of restaurants such as "*special French fries*." The d of e is the specific address of the restaurant.

Celebrity: The q in this dataset is to find some people with a certain characteristic, *e.g.*, "*senior engineer*." The entity descriptions are also informative. Some of them are considered as detailed information about job information, and others may reflect some experience of e. Also, one of the biggest features of this dataset is that not all e have d. This feature also increases the difficulty of the entity search task.

In the dataset mentioned above, the candidate pool size (*p_size*) is about 100, so it is a challenging setting. In a real entity search system, the search engine can retrieve a few high-quality candidate entities, and the *p_size* is maybe only more than a dozen candidates. Also, vaguely related entities are used as candidate entities on the Chinese dataset. As a result, if a model achieves good performance on those datasets, it will probably be able to do a good job when the *p_size* becomes smaller.

We use two query sets *ListSearch* and *INEX-LD* provide by Balog and Neumayer [30] with a baseline running results of FSDM as the English dataset (as shown in Table 2). The other two sets are not fit into our entity search task because the search task of *SemSearch ES* is retrieving one entity and the queries in *QALD-2* are natural language questions which are different from keyword queries in other datasets.

TABLE 2. Statistics of two query sets in English dataset.

Datasets	Queries	Entities	Vocabulary Size
ListSearch	115	12393	15061
INEX-LD	100	11625	14162

B. EXPERIMENTAL SETUP

1) EVALUATION METRICS

The quality of an entity search model will be evaluated by *Mean Average Precision* (MAP).

$$MAP = \frac{1}{|Q|}\sum_{i=1}^{|Q|} AverP(C_i, A_i) \tag{4}$$

$|Q|$ denotes the total number of entity queries in the test set. $AverP(C, A) = \frac{\sum_{k=1}^{n}(P(k)\cdot rel(k))}{min(m,n)}$ denotes the average precision (AP). k is the rank in the sequence of retrieved candidate entities. m is the number of correct entities. n is the number of retrieved candidate entities. $P(k)$ is the precision at cut-off k in the candidate entity list. $rel(k)$ is an indicator function equaling one if the entity at rank k is a ground truth entity, and zero otherwise. The reason why we use AP is that the precision rate just considers the number of correct items in the return list, without taking into account the order between items. For an entity search system, the candidate entities must be returned orderly, and the most relevant entity should be ranked in the front of the return list. Another two evaluation metrics we use are *Top-1 accuracy* and *Hit@10* for the Chinese dataset. For a high availability entity search system, users are often concerned about whether the first entity (*Top-1*) in the list meets the requirements, and *Hit@10* means the proportion of correct candidate entities ranked in

[2][Online]. Available: https://github.com/eshijia/baidu_entity_dataset
[3][Online]. Available: http://krisztianbalog.com/resources/sigir-2013-dbpedia/

TABLE 3. **Experimental results of Chinese datasets with different embedding strategies. The significance is tested against the baseline model. Best scores are in boldface.**

Datasets	**tvShow**			**movie**			**restaurant**			**celebrity**		
Metrics	*Top-1*	*Hit@10*	*MAP*	*Top-1*	*Hit@10*	*MAP*	*Top-1*	*Hit@10*	*MAP*	*Top-1*	*Hit@10*	*MAP*
W2V	0.305	0.615	0.201	0.404	0.764	0.299	0.352	0.799	0.250	0.531	0.723	0.311
pFME-1	0.331$^{\vartriangle}$	0.689$^{\vartriangle}$	0.242$^{\vartriangle}$	0.465	0.827$^{\vartriangle}$	0.359$^{\vartriangle}$	0.409	0.885$^{\vartriangle}$	0.282$^{\vartriangle}$	0.444	0.788$^{\vartriangle}$	0.324
pFME-2	0.369$^{\blacktriangle}$	0.748$^{\blacktriangle}$	0.273$^{\blacktriangle}$	0.539$^{\vartriangle}$	0.857$^{\vartriangle}$	0.401$^{\blacktriangle}$	0.451$^{\vartriangle}$	0.910$^{\blacktriangle}$	0.306	0.457	0.821$^{\blacktriangle}$	0.359$^{\vartriangle}$
pFME-3	0.343$^{\vartriangle}$	0.712$^{\blacktriangle}$	0.260$^{\blacktriangle}$	0.505$^{\vartriangle}$	0.839$^{\vartriangle}$	0.384$^{\vartriangle}$	0.422	0.900$^{\blacktriangle}$	0.293$^{\vartriangle}$	0.452	0.807$^{\vartriangle}$	0.346
FME-2	0.255	0.613	0.202	0.399	0.774	0.315	0.391	0.878$^{\vartriangle}$	0.286$^{\vartriangle}$	0.338	0.676	0.241
pERE-1	**0.380**$^{\blacktriangle}$	**0.795**$^{\blacktriangle}$	**0.286**$^{\blacktriangle}$	**0.645**$^{\blacktriangle}$	**0.920**$^{\blacktriangle}$	**0.477**$^{\blacktriangle}$	0.532$^{\blacktriangle}$	0.918$^{\blacktriangle}$	**0.309**$^{\blacktriangle}$	0.143	0.562	0.163
pERE-2	0.332$^{\vartriangle}$	0.790$^{\blacktriangle}$	0.270$^{\blacktriangle}$	0.634$^{\blacktriangle}$	0.914$^{\blacktriangle}$	0.461$^{\blacktriangle}$	**0.534**$^{\blacktriangle}$	**0.919**$^{\blacktriangle}$	0.306$^{\blacktriangle}$	0.144	0.566	0.164
pERE-3	0.271	0.785$^{\blacktriangle}$	0.261$^{\blacktriangle}$	0.619$^{\blacktriangle}$	0.889$^{\blacktriangle}$	0.410$^{\blacktriangle}$	0.492$^{\blacktriangle}$	0.899$^{\blacktriangle}$	0.283$^{\vartriangle}$	0.137	0.541	0.152
ERE-2	0.256	0.780$^{\blacktriangle}$	0.257$^{\vartriangle}$	0.608$^{\blacktriangle}$	0.811	0.400$^{\blacktriangle}$	0.481$^{\blacktriangle}$	0.874$^{\vartriangle}$	0.272	0.129	0.522	0.145
pTDE-1	0.357	0.765$^{\blacktriangle}$	0.272$^{\blacktriangle}$	0.549$^{\vartriangle}$	0.877$^{\blacktriangle}$	0.408$^{\blacktriangle}$	0.406	0.893$^{\blacktriangle}$	0.307$^{\blacktriangle}$	0.471	**0.853**$^{\blacktriangle}$	**0.385**$^{\blacktriangle}$
pTDE-2	0.374$^{\blacktriangle}$	0.776$^{\blacktriangle}$	0.279$^{\blacktriangle}$	0.535$^{\vartriangle}$	0.842$^{\vartriangle}$	0.391$^{\blacktriangle}$	0.382	0.870$^{\vartriangle}$	0.281$^{\vartriangle}$	0.454	0.851$^{\blacktriangle}$	0.382$^{\blacktriangle}$
pTDE-3	0.367$^{\blacktriangle}$	0.757$^{\blacktriangle}$	0.269$^{\blacktriangle}$	0.521$^{\vartriangle}$	0.834$^{\vartriangle}$	0.374$^{\vartriangle}$	0.352	0.861$^{\vartriangle}$	0.265	0.452	0.839$^{\blacktriangle}$	0.376$^{\blacktriangle}$
TDE-2	0.267	0.642$^{\vartriangle}$	0.202	0.419	0.805	0.323	0.328	0.847	0.259	0.297	0.662	0.236

the top 10. For the English datasets, our model is evaluated by MAP@100, P@10, and P@20 following previous work as [25] and [26]. We use 5-fold cross validation,[4] and the reported results are the average of 5 folds.

2) IMPLEMENTATION

Our entity search model in this paper was built from scratch using Python with Keras,[5] and all experiments were processed in a Tesla k20C GPU device.

We used a large-scale corpus[6] crawled from Baidu Encyclopedia (similar to Wikipedia) to learn the pre-trained Chinese word embeddings with word2vec,[7] and the embedding size (n) was 300. Baidu Encyclopedia contains a large number of Chinese entities, and training the Chinese word embedding with that corpus can make the pre-trained word embedding cover much semantic information about the entities in training set. For the English version, we directly used the embeddings trained on the part of Google News dataset. We tried several margin values, such as 0.5, 0.2, 0.05, 0.02, and we chose 0.02 at last as it produced the best results with the cross-validation. We selected the dropout rate of the dropout layer from 0.5 and 0.25. We trained our models with Adam as the final optimization strategy. The learning rate (α) was 0.001. In the following, there will be a detailed comparison of different optimization strategies. The batch size (b_size) was 64. The hyper-parameters were tuned based on measuring the validation result.

C. RESULTS

In addition to the contrast between different embedding strategies, we also evaluated the vanilla word embedding method with Chinese dataset. The baseline utilizes vanilla word embeddings (W2V) trained by *word2vec* [4], *i.e.*, we directly use the pre-trained word embeddings of v, and pairwise calculate the cosine similarity between the embeddings in q and c_ent, then select the maximum similarity as the matching score between q and c_ent. As for the English dataset, we compared our model with the FSDM and previous state-of-art learning to rank models.

1) ENTITY EMBEDDING

Our proposed embedding strategies can make the entities have effective semantic information. It is too noisy to plot all the entity distributed representations at the same time. For example, Figure 4 shows two parts of entity vectors visualized by t-SNE [31] with the pTDE-1 embedding strategy.

We can find that entities with similar semantic meaning are well distributed in the same region. Also, even for similar entities, they can be distinguished on fine granularity. For example, in Figure 4b, the *Country* entities are distributed in the same region. Moreover, Asian countries and European countries are divided into different fine-grained regions.

2) PERFORMANCE ANALYSIS

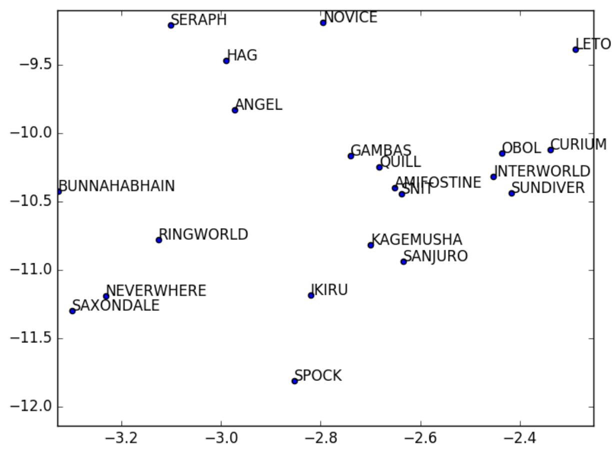
Table 3 shows the detail results of different entity search models on Chinese dataset. In Table 3, the model named with the prefix of 'p' indicates that it uses the pre-trained word vectors, and with a suffix of '1' means the dropout rate of its dropout layer is 0.5, '2' means 0.25 and '3' means it does not use the dropout layer. Therefore, compared to vanilla W2V model, the proposed models with three embedding strategies are more suitable for the entity search task with higher *MAP*. For **tvShow**, **movie** and **restaurant** collections where *des* is relatively simple and lack of information, the ERE models achieve the best performance, and it can prove that if the description doesn't contain much semantic information the proposed model ERE can still have a good performance. The experiment results show that the TDE models achieve the best performance for **celebrity** collection, and it verifies that the distributed representation of q, ent, and des can be

[4][Online]. Available: http://scikit-learn.org/stable/modules/cross_validation.html

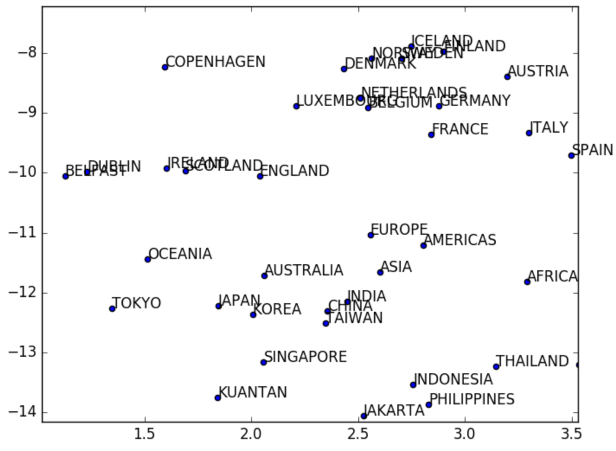
[5][Online]. Available: https://keras.io

[6]The text corpus size is about 960 GB.

[7][Online]. Available: https://code.google.com/archive/p/word2vec



(a)



(b)

FIGURE 4. **The visualization of the fine-tuned entity embedding. (a) The distribution of entities with similar semantic meaning. (b) Similar entities can be distinguished on fine granularity.**

calculated by each other in the low-dimensional vector space, and the description contain much semantic information can benefit the TDE model.

Also, we can make following research results from Table 3. The dropout layer can effectively improve the performance of models (*e.g.*, pERE-2 vs. pERE-3). Without the pre-trained word embeddings (*e.g.*, FME-2, ERE-2 or TDE-2), the overall performance of the model will naturally decrease. The fact that the FME models do not achieve excellent performance proves that methods used in the answer selection task cannot directly apply to the entity search task, and we should consider the short text property of named entities and entity descriptions to solve the entity search task.

Table 4 shows the experimental results of English datasets with the proposed models. The description of the entities in the English dataset is available in DBpedia. Therefore, we can further prove that the description would be helpful for entity ranking. As follows from Table 4, the proposed models outperform previously learn to rank models on all query sets. Besides, the performance of pTDE-1 yields about 2%-3% absolute improvement over the pERE-1 model. Because we want to make a fair comparison, we also take

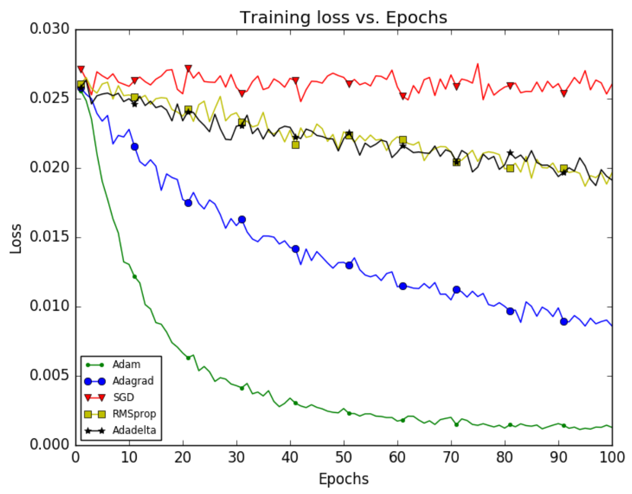


FIGURE 5. **Training loss versus epochs with various optimizers.**

the top 100 entities returned by FSDM as the *c_ent* to our proposed model. However, FSDM itself may not retrieve correct entities. Therefore, we directly apply the representation models (pERE-1* and pTDE-1*) to the raw data, and the evaluation results are further improved.

Examples of the entity search results for pERE-1* and pTDE-1* are compared in Table 5. The entity description for entities in the examples is shown in Table 6. From the results, we have the following conclusion. 1) The entity description is useful to enrich the semantic information of a candidate entity, and it can benefit the entity search model. 2) If the length of an entity search query is too short, the entity description has no significant effect on the results. For example, the entity search models do not understand the specific meaning of the entity search query "Eiffel", and cannot determine the "Eiffel" means the landmark of Paris or the Italian musical group *Eiffel 65*. Therefore, the consideration of incorporating the query expand methods to make the entity search query contain more information will be a subsequent work.

3) OPTIMIZER COMPARISION

For the training process of our models, the optimizer had a significant influence on the convergence of learning models. Different optimizers fit different training tasks. Figure 5 shows the trends of the loss value in *ERE-2 model* with different popular optimizers. The performance of vanilla SGD [32] is poor. Within 100 epochs, there was no effective reduction of the loss value. Adagrad [32] adapted the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters, and it had a better convergence effect in this entity search task. Adadelta [33] and RMSprop [34] were both extensions of Adagrad, and they sought to reduce the aggressive, monotonically decreasing learning rate. The difference was that RMSprop divided the learning rate by an exponentially decaying average of squared gradients. However, there was no significant difference in the convergence

TABLE 4. Comparison between the pERE-1 model and several learning to rank methods. Significance is tested against the FSDM model. The numbers in parentheses show the relative improvements. Best scores are in boldface.

	ListSearch					
	MAP@100		P@10		P@20	
FSDM	0.203	–	0.256	–	0.203	–
ELR	0.197	(-2.96%)	0.239	(-6.64%)	–	–
RankSVM	0.224	(+10.3%)	0.303	(+18.7%)	0.235	(+15.9%)
Coordinate Ascent	0.225	(+10.5%)	0.300	(+17.3%)	0.229	(+12.9%)
pFME-1	0.224	(+10.3%)	0.313	(+22.3%)	0.236	(+16.2%)
pERE-1	0.257	(+26.6%)	0.350	(+36.7%)	0.266	(+31.0%)
pTDE-1	**0.260**$^{\blacktriangle}$	(+28.1%)	**0.352**$^{\blacktriangle}$	(+37.5%)	**0.267**$^{\blacktriangle}$	(+31.5%)
pERE-1*	0.262	(+29.1%)	0.359	(+40.2%)	0.270	(+33.0%)
pTDE-1*	**0.271**$^{\blacktriangle}$	(+33.5%)	**0.360**$^{\blacktriangle}$	(+40.6%)	**0.275**$^{\blacktriangle}$	(+35.5%)

	INEX-LD					
	MAP@100		P@10		P@20	
FSDM	0.111	–	0.263	–	0.214	–
ELR	0.133	(+19.8%)	0.233	(-11.4%)	–	–
RankSVM	0.126	(+12.9%)	0.282	(+7.2%)	0.231	(+7.7%)
Coordinate Ascent	0.121	(+8.9%)	0.275	(+4.6%)	0.224	(+4.4%)
pFME-1	0.162	(+45.9%)	0.290	(+10.3%)	0.245	(+14.5%)
pERE-1	0.189	(+70.3%)	0.352	(+33.8%)	0.287	(+34.1%)
pTDE-1	**0.193**$^{\blacktriangle}$	(+73.8%)	**0.359**$^{\blacktriangle}$	(+36.5%)	**0.292**$^{\blacktriangle}$	(+36.4%)
pERE-1*	0.194	(+74.8%)	0.360	(+36.9%)	0.292	(+36.4%)
pTDE-1*	**0.199**$^{\blacktriangle}$	(+79.3%)	**0.368**$^{\blacktriangle}$	(+39.9%)	**0.298**$^{\blacktriangle}$	(+39.3%)

TABLE 5. Examples of the entity search results for pERE-1* and pTDE-1*. Correct entities are in bold.

Entity search queries	Top-3 results of pERE-1*	Top-3 results of pTDE-1*
composer with a professor title	Yong Hou **Yanling Wang** Lee Gook Ju	**Peng Guan** **Yanling Wang** Jianping Cao
Indian food	**Gujarati cuisine** **Paneer** Manju Malhi	**Tamil cuisine** **Goan cuisine** **Puri (food)**
Eiffel	**Eiffel Tower** **Eiffel company** Eiffel 65	**Eiffel Tower** Lisaac Move Your Body

TABLE 6. Entity description for the entities in the examples.

Entity	Entity description
Yong Hou	associate professor in Southwest University
Peng Guan	composer
Yanling Wang	Professor in college of liberal arts, Jilin University
Lee Gook Ju	(no entity description in the dataset)
Jianping Cao	plastic surgery expert
Gujarati cuisine	Gujarati cuisine refers to the cuisine of Gujarat, a state in western India
Tamil cuisine	Tamil Nadu is famous for its deep belief that serving food to others is a service to humanity, as it is common in many regions of India
Paneer	Paneer is a fresh cheese common in South Asia, especially in Indian, Pakistani and Bangladeshi cuisines
Goan cuisine	Goan cuisine consists of regional foods popular in Goa, an Indian state located along India's west coast on the shore of the Arabian Sea.
Manju Malhi	Manju Malhi is a British-born chef and food writer
Puri (food)	Puri is an unleavened deep-fried Indian bread, commonly consumed on the Indian subcontinent.
Eiffel Tower	The Eiffel Tower is a wrought iron lattice tower on the Champ de Mars in Paris, France.
Eiffel company	Eiffel is part of the Eiffage group and the descendant of the engineering company founded by Gustave Eiffel, designer of the Eiffel Tower.
Lisaac	The developers of Lisaac included features such as dynamic inheritance from Self and contract programming from Eiffel.
Eiffel 65	Eiffel 65 is an Italian musical group consisting of Jeffrey Jey, Maurizio Lobina and Gabry Ponte.
Move Your Body	"Move Your Body" is a song by Italian group Eiffel 65. It was released as the second single from their album Europop.

effects of Adadelta and RMSprop. Adam [6] also calculated adaptive learning rates for each parameter. Unlike Adadelta and RMSprop update the learning rate just based on the exponentially decaying average of *past squared gradients*, Adam also kept an exponentially decaying average of *past gradients* that similar to momentum, and it got the highest convergence rate. Therefore we could achieve better training results with a shorter training time, which was the main reason that we chose the Adam optimizer. It should be noted that we must regularly assess the performance of the validation set to prevent overfitting with the fast optimizer.

V. CONCLUSION

In this paper, we study the problem of entity search by employing representation learning methods. With three embedding strategies, our proposed model can adapt to different kinds of entity query tasks. Experimental results on four Chinese and two English entity collections validate the superiority of the proposed models, and we also give an analysis about how to choose a proper optimizer to train the representation models.

The core idea of our models is to consider the relationships among named entities, entity descriptions and entity search queries in the low-dimensional vector space. Also, we pay attention to the short text features of them. The overall framework in this paper is language independent, and it can be easily generalized to other similar tasks. There are many potential future research directions for this work. A significant trend is that combining with more external data sources, such as knowledge base, to obtain more entity features, thus improving the overall performance of the models. Our work is a comprehensive experimental study of utilizing representation learning methods to solve the entity search task.

REFERENCES

[1] Y. Yang and J. Tang, "Beyond query: Interactive user intention understanding," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2015, pp. 519–528.

[2] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Clustering user queries of a search engine," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 162–168.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (Sep. 2013). "Efficient estimation of word representations in vector space." [Online]. Available: https://arxiv.org/abs/1301.3781

[5] M. Feng, B. Xiang, M. R. Glass, L. Wang, and B. Zhou, "Applying deep learning to answer selection: A study and an open task," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand. (ASRU)*, Dec. 2015, pp. 813–820.

[6] D. P. Kingma and J. Ba. (Dec. 2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[7] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. (Apr. 2016). "Neural architectures for named entity recognition." [Online]. Available: https://arxiv.org/abs/1603.01360

[8] A. Bordes, S. Chopra, and J. Weston. (Sep. 2014). "Question answering with subgraph embeddings." [Online]. Available: https://arxiv.org/abs/1406.3676

[9] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. ICML*, vol. 14. 2014, pp. 1188–1196.

[10] S. Lai, K. Liu, S. He, and J. Zhao, "How to generate a good word embedding?" *IEEE Intell. Syst.*, vol. 31, no. 6, pp. 5–14, Nov./Dec. 2015.

[11] J. Y. Lee and F. Dernoncourt. (Mar. 2016). "Sequential short-text classification with recurrent and convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1603.03827

[12] C. N. D. Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proc. COLING*, 2014, pp. 69–78.

[13] C. Van Gysel, M. de Rijke, and E. Kanoulas, "Learning latent vector spaces for product search," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, 2016, pp. 165–174.

[14] M. Tan, B. dos Xiang, and B. Zhou. (Nov. 2015). "LSTM-based deep learning models for non-factoid answer selection." [Online]. Available: https://arxiv.org/abs/1511.04108

[15] D. Cohen and W. B. Croft, "End to end long short term memory networks for non-factoid question answering," in *Proc. ACM Int. Conf. Theory Inf. Retr.*, 2016, pp. 143–146.

[16] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proc. Aistats*, vol. 5. 2005, pp. 246–252.

[17] A. Mnih and Y. W. Teh. (Jun. 2012). "A fast and simple algorithm for training neural probabilistic language models." [Online]. Available: https://arxiv.org/abs/1206.6426

[18] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 160–167.

[19] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng. (Feb. 2016). "Text matching as image recognition." [Online]. Available: https://arxiv.org/abs/1602.06359

[20] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2042–2050.

[21] Z. Lu and H. Li, "A deep architecture for matching short texts," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1367–1375.

[22] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng. (Nov. 2015). "A deep architecture for semantic matching with multiple positional sentence representations." [Online]. Available: https://arxiv.org/abs/1511.08277

[23] A. Severyn and A. Moschitti, "Learning to rank short text pairs with convolutional deep neural networks," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2015, pp. 373–382.

[24] D. Graus, M. Tsagkias, W. Weerkamp, E. Meij, and M. de Rijke, "Dynamic collective entity representations for entity ranking," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, 2016, pp. 595–604.

[25] J. Chen, C. Xiong, and J. Callan, "An empirical study of learning to rank for entity search," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2016, pp. 737–740.

[26] N. Zhiltsov, A. Kotov, and F. Nikolaev, "Fielded sequential dependence model for ad-hoc entity retrieval in the Web of data," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2015, pp. 253–262.

[27] F. Hasibi, K. Balog, and S. E. Bratsberg, "Exploiting entity linking in queries for entity retrieval," in *Proc. ACM Int. Conf. Theory Inf. Retr.*, 2016, pp. 209–218.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] M. Fan, Q. Zhou, E. Chang, and T. F. Zheng, "Transition-based knowledge graph embedding with relational mapping properties," in *Proc. 28th Pacific Asia Conf. Lang. Inf. Comput.*, 2014, pp. 328–337.

[30] K. Balog and R. Neumayer, "A test collection for entity search in DBpedia," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2013, pp. 737–740.

[31] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[32] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.

[33] M. D. Zeiler. (Dec. 2012). "ADADELTA: An adaptive learning rate method." [Online]. Available: https://arxiv.org/abs/1212.5701

[34] T. Tieleman and G. Hinton, "Lecture 6.5-Rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA, Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

SHIJIA E (S'17) received the B.S. degree in computer science and technology from Tongji University, China, in 2014, where he is currently pursuing the Ph.D. degree with the College of Electronics and Information Engineering. His research interests include representation learning, knowledge graph mining, and distributed computing.

YANG XIANG received the Ph.D. degree in management science and engineering from the Harbin Institute of Technology in 1999. He is currently a Professor with the Department of Computer Science and Technology, Tongji University. He has published more than 100 papers in international journals and conferences, including the *Expert Systems with Applications*, *Science China Information Sciences*, and the *Chinese Journal of Electronics*. He has published four books on intelligent decision support system in complex problems. His research interests include data warehousing, data mining, intelligent decision support system, service computing, and e-commerce.