Cluster Comput
DOI 10.1007/s10586-017-1273-x

A joint deep model of entities and documents for cumulative citation recommendation

Lerong Ma[1,2] · **Dandan Song**[1] · **Lejian Liao**[1] · **Yao Ni**[1]

Received: 29 June 2017 / Revised: 27 September 2017 / Accepted: 12 October 2017

Abstract Knowledge bases (such as Wikipedia) are valuable resources of human knowledge which have contributed to various of applications. However, their manual maintenance makes a big lag between their contents and the up-to-date information of entities. Cumulative citation recommendation (CCR) concentrates on identifying worthy-citation documents from a large volume of stream data for a given target entity in knowledge bases. Most previous approaches first carefully extract human-designed features from entities and documents, and then leverage machine learning methods such as SVM and Random Forests to filter worthy-citation documents for target entities. There are some problems in handcraft features for entities and documents: (1) It is an empirical process that requires expert knowledge, thus cannot be easily generalized; (2) The effectiveness of humanly designed features has great effect on the performance; (3) The implementation of the feature extraction process is resource dependent and time-consuming. In this paper, we present a Joint Deep Neural Network Model of Entities and Documents for CCR, termed as DeepJoED, to identify highly related documents for given entities with several layers of neurons, by automatically learn feature extraction of the entities and documents, and train the networks in an end-to-end fashion.An extensive set of experiments have been conducted on the benchmark dataset provided in the Text REtrieval Conference (TREC) Knowledge base acceleration (KBA) task in 2012. The results show the model can bring a significant improvement relative to the state-of-the-art results on this dataset in CCR.

Keywords Knowledge base acceleration · Cumulative citation recommendation · Word embedding · Convolution Neural Networks · Latent semantic representations

1 Introduction

Knowledge Bases (KBs), like Wikipedia[1] and Satori[2], have been successfully leveraged in various applications including entity linking [1], query expansion [2], knowledge graph [3], question answering [4], entity retrieval [5] and recommender systems [6]. Keeping the contents of KBs timely is crucial to these applications. However, it is very hard for most KBs to be up-to-date owing to their manual maintenance by human editors. There is a median time lag of one year between the publication date of a news article and the date that the news article is edited into a Wikipedia profile [7]. This time lag would be significantly reduced if documents with high relevance to the target entity in the KBs could be detected automatically as soon as the documents are published online and then recommended to the editors. This task is studied as knowledge base acceleration-cumulative citation recom-

✉ Dandan Song
sdd@bit.edu.cn

Lerong Ma
malerong2008@163.com

Lejian Liao
Liaolj@bit.edu.cn

Yao Ni
2120161025@bit.edu.cn

[1] Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

[2] College of Mathematics and Computer Science, Yan'an University, Yan'an 716000, China

[1] https://www.wikipedia.org/

[2] http://searchengineland.com/library/bing/bing-satori

Published online: 17 October 2017

mendation (KBA-CCR) by the Text REtrieval Conference (TREC). Formally, given a set of knowledge base entities, CCR is to automatically detect worthy-citation documents from a large volume of stream data.

Many relevant supervised models (e.g., classification, learning to rank) have been used to solve the CCR task, and achieved promising performances [8–10] by carefully exploiting hand-designed features from entities and documents. Balog [9] has conducted an extensive set of experiments on the TREC-KBA-2012 dataset with a total of 68 independent features involving entity features, document features, temporal features, and document-entity features. These approaches first extract human designed features from entities and documents, and then fed these features to the machine learning algorithm. Obviously, this way suffers from the following limitations: (1) The choice of features is an empirical process by expert design with low generalization ability. (2) The effectiveness of humanly designed features would have great effect on the performance; (3) The realization of feature extraction is always resource dependent and time-consuming to be implemented.

To address the above issues, we present a Joint Deep Neural Network Model of Entities and Documents for CCR, named as DeepJoED, to learn distributed representations of entities and documents automatically in an end-to-end fashion. This approach is inspired by deep learning successfully applying to various natural language processing applications. Our DeepJoED takes the raw texts of entities and documents as input, learns latent features for entities and documents jointly using two coupled neural networks with several layers of neurons for feature representations, couples the latent representations of entities with those of documents by a hidden layer to predict the worthy-citation level of the document for the target entity, and trains the networks by backpropagation in an end-to-end fashion.

We have conducted an extensive set of experiments on the benchmark TREC-KBA-2012 dataset. The results reveal the model achieves significant performance gains relative to the state-of-the-art results on this dataset in the CCR task.

The key contributions of this paper are listed in the following:

- To the best of our knowledge, the proposed Joint Deep Neural Network Model is the first model to deal with the CCR task using deep neural networks. In the network structure, DeepJoED jointly models entities and documents with their raw texts as input by two coupled neural networks. Then the interaction layer couples the two parallel networks and the output layer further computes their relevance scores.
- With simple raw texts inputs, our DeepJoED model doesn't require any human interpretation for feature extraction. It represents entities and documents as word-embeddings sequences using either pre-trained Word2vec model or random initialized vectors, and trains the model by backpropagation. It provides an end-to-end approach to CCR task. Furthermore, the model can be extended for online learning scenarios where the model needs to be updated continuously with new data.
- We conduct a series of experiments of the proposed DeepJoED model on the TREC-KBA-2012 dataset, the experimental results show that the model outperforms the state-of-art results in the CCR task.

2 Related work

2.1 Cumulative citation recommendation

TREC launched the KBA-CCR (also known as CCR) track since 2012. Participates and researchers treat CCR task as either a ranking problem [5,11] or a classification problem [12–14]. Balog and Ramampiaro [9] conducted an extensive experiments using classification methods and learning to rank methods on the TREC-KBA-2012 dataset with a total of 68 human-selected features regarding to entities and documents. These methods first extract various human-selected features of entities and documents, and then the extracted features are fed to a powerful machine learning algorithm. Unlike the previous methods that require firstly extracting various hand-crafted features, we present a joint deep model of entities and documents for CCR that takes the raw texts of entities and documents as input, and trains the model by backpropagation in an end-to-end fashion.

2.2 Deep learning for NLP and IR

Thanks to their ability to learn complex representations of different data in different tasks [15] with the same form of real valued vectors, deep learning approaches have been successfully used in natural language processing (NLP) tasks, information retrieval (IR) applications, malware detection [16] and so on. The foundation of deep learning architecture for NLP and IR is distributed word representations in semantic vectors space [17], where each word is represented with a real valued vector, named as word embedding. A word is previously represented as one-hot vector by indexing words into a vocabulary, but word embeddings are learned by projecting words onto a low dimensional and dense vector space that encodes both semantic and syntactic feature of words.

Using word embeddings, a variety of models have been presented to learn the composition of words to construct phrase, sentence and document representations. Most approaches fall into three types: sequence models, convolutional neural networks (CNN) models and dependency based neural network.

Sequence models such as recurrent neural network (RNN) and Long short-term memory networks (LSTM) have emerged as an effective and scalable model for several sequential data related learning problems [18]. They build sentence (also called short texts) representations in an order-sensitive way. For instance, due to its ability to capture long-distance dependencies, LSTM has re-emerge as popular choice for many sequence-modeling tasks like language modeling, machine translation [19], natural language generation [20] and so on. The central idea behind LSTM is a memory cell which captures information over time, and non-linear gating units which regulate the information flow into and out of the cell, so the output of last state of the memory cell is used as the final sequence representations, such as a sentence or a document. The researchers have presented more sophisticated types of RNN to deal with more complex sequence problems such as Bidirectional RNN, Deep RNN, Multilayer RNN [19].

In recent years, CNN-Based models have demonstrated remarkable performances on sentence modeling and classification tasks. Leveraging convolution operators, these models can extract feature from variable-length phrases corresponding to different filters. For example, Collobert et al. [21] propose a unified neural network architecture and learning algorithm to deal with various NLP tasks. The network takes the sentences as inputs, and then learns feature extraction using several layers of neural networks including word-embedding, convolution units, max-pooling units, linear and non-linear units. Kim [22] presents a simple convolutional neural networks (CNN) with one layer of convolution on top of word vectors for text classification. Kalchbrenner et al. [23] propose a Dynamic Convolutional Neural Network (DCNN) to construct hierarchical features of sentences by one-dimensional convolution and dynamic k-max pooling. DCNN obtains high performance in sentiment prediction, question classification, and Twitter sentiment prediction. Johnson and Zhang [24] propose a Deep Pyramid Convolutional Neural Networks for text Categorization, called deep pyramid CNN (DPCNN). When discrete text is converted to continuous representation, the DPCNN architecture simply alternates a convolution block and a downsampling layer over and over, inducing a deep network in which internal data size shrinks in a pyramid shape. The DPCNN with 15 weighted layers achieves good performance on six benchmark datasets for sentiment classification and topic classification. CNN-Based models also have been successfully employed to conduct semantic matching in web search for document retrieval [25–27], and significant improvement in relevance have been observed.

Dependency based neural network utilizes CNN and RNN to build up the architecture for specific tasks, which can fully exploit CNN and RNN. Zhang et al. [28] present dependency sensitive convolutional neural networks (DSCNN) as a general purpose classification system for both sentence and documents. DSCNN consists of a convolutional layer built on top of long short-term memory (LSTM) networks. For a single sentence, the LSTM network processes the sequence of word embeddings to capture long-distance dependencies within the sentence. The hidden states of the LSTM are extracted to form the low-level representation, and a convolutional layer with variable-size filter and max-pooling operators follow to extract task-specific features for classification purposes. For document modeling, DSCNN first applies independent LSTM networks to each subsentence, then a second LSTM layer is added between the first LSTM layer and the convolutional layer to encode the dependency across different sentences. Lee and Dernoncourt [29] introduce a model based on RNN and CNN for sequential short-text classification. The model consists of two parts. The first part generates a vector representation for each short text using either the RNN or CNN architecture. The other classifies the current short text based on the vector representations of the current as well as a few preceding short texts.

Inspired by these deep learning models, our proposed model integrates two parallel deep networks for entities and documents representations based on word-embedding, convolution and max-pooling, followed by feature concatenation and correlation calculation layers.

3 Problem statement

We consider CCR as a binary classification problem that treats the relevant entity-document pairs as positive instances and irrelevant ones as negative instances. Many probabilistic classification approaches generally can be classified into two categories in the literature: generative models and discriminative models. Discriminative models have attractive theoretical properties [30] and generally perform well relative to their generative counterparts in the field of information retrieval [31,32]. Therefore, we adopt discriminative probabilistic models to model the probability of each entity-document pair using deep neural networks in this paper.

Suppose we have an entity e in a set of knowledge base entities E, and a document d from a document collection C, where $e = \{w_1, w_2, \ldots, w_m\}$ consisting of m word sequences, and $d = \{t_1, t_2, \ldots, t_n\}$ including n word sequences. Our objective is to estimate the relevance between the document d and the given entity e. In other words, we need to estimate the conditional probability of relevance $P(r|e, d)$ regarding an entity-document pair (e, d), where $r \in \{1, 0\}$, 1 indicates the entity-document pair (e, d) a positive instance, and 0 indicates it a negative instance. In this paper, we use a joint deep neural networks of entities and doc-

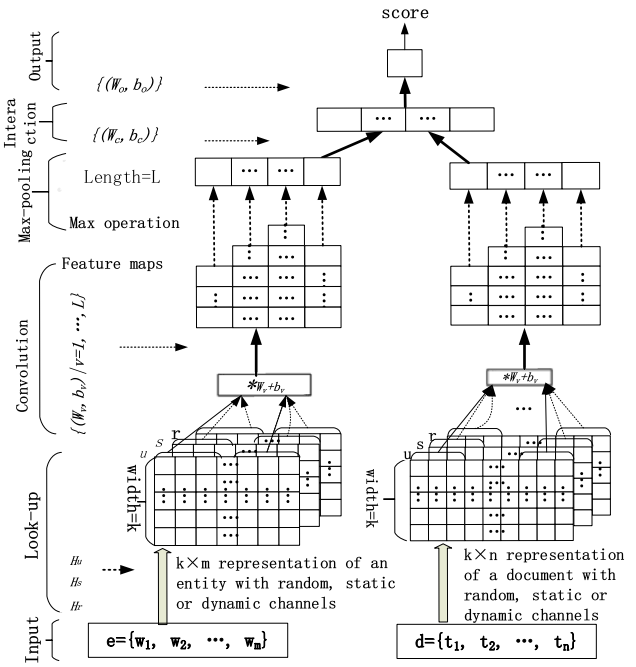


Fig. 1 The architecture of the proposed model. Two parallel networks coupled at the **Interaction** layer, and the relevance score are output at the top of the architecture

uments to model $P(r|e, d)$ in the following in an end-to-end fashion.

4 Methodology

DeepJoED models the relevant of entities and documents using their raw texts. It is a convolutional neural network (CNN) based model consisting of two parallel neural networks, coupled to each other in an interaction layer, and output the relevant scores of an entity and a document at the top of DeepJoED. The networks are trained in a joint manner on a training dataset with minimum loss. The training dataset is composed of entity-document pairs where the entity is from a set of target entities and the document comes from a stream corpus. DeepJoED is summarized in Fig. 1. We will describe DeepJoED in details in this section.

4.1 Architecture of DeepJoED

The architecture of DeepJoED for CCR is shown in Fig. 1. The model consists of two parallel neural networks sharing weights and biases, and coupled in the interaction layer. One network is for entities (Net_e) and the other is for documents (Net_d). For any of entity-document pairs, texts of the entity and document are fed to Net_e and Net_d respectively as input, and corresponding relevance score of the entity-document pair is computed as output.

- In the first layer, denoted as the *Input* layer, the raw word sequence of an entity-document pair are fed to our architecture as sequence indices taken from a finite dictionary D, which was built on the basis of the word set of target entities and the stream corpus. Note that the dictionary D contains a special word as 'unknown', and words out-of-dictionary will be mapped to 'unknown'.
- In the second layer, denoted as the *Look-up* layer, the entity and document are represented as two corresponding matrices of word embeddings by a lookup table operation.
- Next two layers are two layers commonly used in the CNN models—the *Convolution* layer and *Max-pooling* layer. These two layers discover multiple levels of features underling the entity and document.
- The *Interaction* layer is added on the top of the two parallel networks to make latent factors of the entity and document to be combined in the same semantic space. The output of this layer is fed to the top layer.
- The top layer, denoted as the *Output* layer, computes the relevance scores of the entity-document pair as output, by a fully-connected network.

In the following subsections, we will describe each layer in detail. As the two networks Net_e and Net_d are only different in inputs, we focus on explaining the structure of Net_e, the *Interaction* layer, and the *Output* layer.

4.2 Input layer

We first pre-process the dataset of all target entity profiles and all documents from the stream corpus to build a finite dictionary D containing most common words. The first word in the dictionary D is 'unknown', and its index is 0. All words out-of-dictionary are replaced by 'unknown'. In the Input layer, the two networks Net_e and Net_d take the sequence indices of the raw texts of an entity e and a document d through looking up the dictionary D, respectively. Formally, we denote $e = \{w_1, w_2, \ldots, w_m\}$ and $d = \{t_1, t_2, \ldots, t_n\}$, where w_i $(i = 1, \ldots, m)$ and t_j $(j = 1, \ldots, n)$ are indices in the dictionary D, m and n are lengths of the entity e and the document d respectively shown in Fig. 1.

4.3 Look-up layer

Word embedding A word embedding $g : D \rightarrow H$ is a parameterized function that map any of words in the dictionary to the $k-dimensional$ real value vector representation, known as word embedding. D is the dictionary of words that occur in the entity profiles and documents corpus with the first word 'unknown', and $H \in R^{k\times|D|}$ is a matrix with real-valued parameters to be learned. Each column of H (i.e. $[H]_i$) is a vector with real values corresponding to the word

in D (i.e. D_i), and the length of the dictionary D is equal to the number of columns of H. For simplicity, however, D always uses indices of words in the dictionary D to represent the words. Therefore $g : D \rightarrow H$ is the function that projects any index of word in D to the $k - dimensional$ real value vector. In this paper, we identify H in two ways. One is that we use the Word2Vec [33] method to learn word representations from the dataset of all target entity profiles and all documents in the stream corpus. During training process of the networks, we denote the learned H as H_u if H will be updated, otherwise as H_s. The other way is that we randomly initialize H, and then H will be learnt during the training process. In this case, we denote H as H_r. *Note that* H_u,H_s and H_r are regarded as channels similar to images. In our experiments, we conducted experiments with one channel or two channel combinations of word vectors over the set of $\{H_u, H_s, H_r\}$.

Entity and document representations Let x_i and $y_j \in R^k$ be k-dimensions word vectors corresponding to the i-th word in an entity profile and the j-th word in a document, respectively. That can be done by looking-up H. Let $[H]_i$ represents the i^{th} column of matrix H. According to $e = \{w_1, w_2, \ldots, w_m\}$ and $d = \{t_1, t_2, \ldots, t_n\}$ defined in Sect. 4.2, we can define an operator Lu, such that

$$\begin{aligned} x_i &= Lu(w_i) = [H]_{w_i}, i = 1, \ldots, m, \\ y_j &= Lu(t_j) = [H]_{t_j}, j = 1, \ldots, n. \end{aligned} \tag{1}$$

Thus, for the entity e and document d, we can produce the following two output matrices:

$$\begin{aligned} x_e &= x_{1:m} = \left([H]_{w_1}[H]_{w_2} \ldots [H]_{w_m}\right), \\ y_d &= y_{1:n} = \left([H]_{t_1}[H]_{t_2} \ldots [H]_{t_n}\right), \end{aligned} \tag{2}$$

where x_e and y_d denote the semantic representations of the entity e and the document d, respectively. Note that H can be replaced by H_r, H_s or H_u. Obviously, this representation retains the orders of words in the entity profile and the document. These two matrices can then be fed to the *Convolution layer*, as will be explained below.

4.4 Convolution and max-pooling layers

In order to use convolution operations over x_e and y_d in (2) to capture local semantic features of the entity and document, we let $x_{i:i+l}$ and $y_{j:j+l}$ refer to the concatenations of $x_i, x_{i+1}, \ldots, x_{i+l}$ and $y_j, y_{j+1}, \ldots, y_{j+l}$ to build up word embedding segment sequences, respectively. More formally,

$$\begin{aligned} x_{i:i+l} &= \left([H]_{w_i}[H]_{w_{i+1}} \ldots [H]_{w_{i+l}}\right), \\ y_{j:j+l} &= \left([H]_{t_j}[H]_{t_{j+1}} \ldots [H]_{t_{j+l}}\right) \end{aligned} \tag{3}$$

are two sub-matrices.

Given a set of filters and bias terms $F = \{(W_v, b_v)|v = 1, \ldots, L\}$, a convolution operation involves a filter (kernel) $W_v \in R^{d_{win} \times k}$ and a bias term $b_v \in R$ from F, which are applied to a window of d_{win} words to produce a local feature. For instance, a local feature e_{l_i} is produced from a window of words $x_{i:i+d_{win}-1}$ of the entity x_e by

$$e_{l_i} = f\left(W_v \cdot x^t_{i:i+d_{win}-1} + b_v\right), \tag{4}$$

where f is a non-linear function such as Rectified Linear Units ($ReLUs$) [34]. This filter is applied to any possible window of words in the entity x_e :

$$\{x_{1:d_{win}}, x_{2:d_{win}+1}, \ldots, x_{m-d_{win}+1:m}\}$$

to compute a feature map

$$e_v = \left[e_{l_1}, e_{l_2}, \ldots, e_{l_{m-d_{win}+1}}\right], \tag{5}$$

where e_v corresponding to the filter W_v and the bias term b_v. Similarly, we can obtain

$$d_v = \left[d_{l_1}, d_{l_2}, \ldots, d_{l_{n-d_{win}+1}}\right], \tag{6}$$

where $d_{l_i}(i = 1, \ldots, n - d_{win} + 1)$ is a local feature of the document y_d by applying the filter W_v and the bias term b_v. Please note that for each window size, it corresponds to L separate filters. For simplicity, we suppose here one window size (i.e. total L filters). In the two channel combination architecture shown in Fig. 1, each filter is applied to both the combined channels similar to the processing of images. We noted that L and d_{win} are hyper-parameters to be learnt during training of the model.

According to the outputs of convolution operation units, we then apply the max pooling operation [21] over the feature maps e_v and d_v. After that, we acquire two fixed size vectors shown in (7) to feed it to the next layer.

$$\begin{aligned} e_{max} &= \left[\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_L\right] \\ d_{max} &= \left[\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_L\right], \end{aligned} \tag{7}$$

where $\hat{e}_v(v = 1, \ldots, L)$ is max value among e_v defined in (5), and $\hat{d}_v(v = 1, \ldots, L)$ is max value among d_v defined in (6). This pooling scheme indicates that we can naturally deal with various length of entities or documents, and acquires fixed-size outputs.

4.5 Interaction layer

Although the two fixed-sized outputs (7) can be viewed as features of the entity e and the document d, they are in dif-

ferent feature spaces and cannot be directly used. Thus, in order to map them into the same feature space, we introduce a hidden layer, called *Interaction* layer, to couple Net_e and Net_d.

Formally, we first concatenate e_{max} and d_{max} (7) into a single vector $z = [e_{max}, d_{max}]$, i.e.,

$$z = \left[\hat{e}_1, \hat{e}_2, ..., \hat{e}_L, \hat{d}_1, \hat{d}_2, ..., \hat{d}_L\right] \tag{8}$$

where the size of z is $2L$, and L is the number of filters. Then, let $W_c \in R^{2L \times L}$ and $b_c \in R^L$, and the output of *Interaction* layer is as follows:

$$I = f\left(W_c^t * z^t + b_c\right) \tag{9}$$

where f is also a non-linear function such as Rectified Linear Units($ReLUs$) [34], and W_c^t and z^t denote the transpose of W_c and z, respectively. These outputs are fed to the next layer of the network.

4.6 Output layer

The *Output* layer computes the relevance score of the entity e and the document d. Formally, let $W_o \in R^{L \times 1}$ and $b_o \in R^1$, and we define the output score as follows:

$$score = W_o^t * I^t + b_o \tag{10}$$

where W_o^t and I^t are the transpose of W_o and I, respectively. According to the $score$, we determine the conditional probability

$$P(r = 1|e, d) = \sigma(score) \tag{11}$$

for the entity-document pair (e, d), where σ is the sigmoid function.

5 Network learning

Because of the non-uniform distribution of positive instances and negative instances in the training set in most cases, we introduce an parameter γ to trade off recall and precision through up- or down-weighting the cost of a positive error relative to a negative error. The objective loss function is the weighted cross entropy of the training data. More specifically, assuming entity-document pairs in training set are represented as $T = \{(e_q, d_q)\}$, and $R = \{r_q\}$ denotes the corresponding relevance judgement ($i.e.$, $+1$ or 0) of (e_q, d_q) where $q = 1, \ldots, N$, and training instances are independently generated. Thus, the weighted cross entropy of the training data is written as follows:

Table 1 Hyperparameters and their definitions

Hyperparameters	Definitions
k	The dimensionality of word embedding
d_{win}	The window size of convolution operation
L	The number of filters for each window size
γ	The weights of positive instances
p	The proportion of dropout
$batch\_size$	The size of batch for training

$$\begin{aligned} L(\theta) = \sum_{q=1}^{N} \Big[& r_q * \left(-log\left(P\left(r_q|e_q, d_q\right)\right)\right) * \gamma \\ & + (1 - r_q) * \left(-log\left(1 - P\left(r_q|e_q, d_q\right)\right)\right) \Big] \end{aligned} \tag{12}$$

where $P(r_q|e_q, d_q) = \sigma(score_q)(q = 1, \ldots, N)$ is the output of the network in (11), and θ denotes the trainable parameters of the network, which are optimised using batch stochastic gradient descent [35].

5.1 Regularization

In order to cope with "overfitting", we leverage dropout [36] on the outputs of the *Max-pooling* layer. Dropout prevents complex co-adaptations of hidden units by randomly dropping out the hidden units with a proportion $p \in [0, 1]$ during forward-backpropagation. More specifically, given

$$\begin{aligned} e_{max} &= \left[\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_L\right] \\ d_{max} &= \left[\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_L\right] \end{aligned}$$

in (7), and $\mathbf{r} \in \{0, 1\}^L$ is a "masking" vector of random variables with each variable having the probability p of being 1 and $1 - p$ of being 0, dropout first perform the following operation

$$\begin{aligned} e_{max} \circ \mathbf{r} &= \left[\hat{e}_1 * r_1, \hat{e}_2 * r_2, \ldots, \hat{e}_L * r_L\right] \\ d_{max} \circ \mathbf{r} &= \left[\hat{d}_1 * r_1, \hat{d}_2 * r_2, \ldots, \hat{d}_L * r_L\right] \end{aligned}$$

where $\circ$ is the element-wise multiplication operator, and then replaces e_{max} and d_{max} with $e_{max} \circ \mathbf{r}$ and $d_{max} \circ \mathbf{r}$ respectively in the further layer of the network. Gradients are backpropagated only through the unmasked units. During test stage, we set the probability p as 1.

5.2 Hyperparameters

In our model, there are six hyperparameters to be determined. These hyper-parameters are summarized in Table 1.

Table 2 Four-point relevance estimation of documents

Type	Definition
Central	Relates directly to the target entity; e.g., the entity is a central figure mentioned in the topics or events.
Relevant	Relates indirectly to the target entity; e.g., refers topics or events that are likely to have an impact on the entity.
Neutral	Not relevant; nothing can be inferred about the target entity, e.g., entity name used in product name.
Garbage	Not relevant; e.g., spam.

Table 3 Number of annotated training and test instances

Type	Central	Relevant	Neutral	Garbage	Total
Train	3525	6500	1757	9382	21171
Test	5256	8426	2470	20439	36591

6 Experiments

6.1 Dataset

We conduct our experiments on the TREC-KBA-2012 dataset, which is a standard test set provided by TREC.[3] The dataset contains a set of target entities and a stream corpus from three sources including *linking*, *social* and *news*.

Entity Set The entity set is composed of 29 Wikipedia items, more specifically, 27 persons and 2 organizations. Each entity is described by a semi-structured article in Wikipedia called profile, and identified uniquely by a `urlname`.

Stream Corpus The stream corpus, spanning the period from October 2011 to April 2012, consists of documents crawled from news, social media, and Linking. The corpus is approximately 9TB of raw texts, and contains 462,676,772 documents. Each document is time-stamped and uniquely identified by a `stream_id` indicating its time of publication. The corpus is split as training and testing instances, with documents from October to December 2011 period as training instances, and the remainder for testing instances. We follow this setup.

Annotation Given a target entity in the set of entities, the relevance of a document in the scream corpus is judged by annotators in terms of a four-point relevance estimation including *Garbage*, *Neutral*, *Relevant* and *Central*, their definitions are listed in Table 2. The detailed annotations of training and testing instances are listed in Table 3.

[3] http://trec-kba.org/kba-ccr-2012.shtml.

6.2 Evaluation scenario

According to the four-point relevance estimation of entity-document pairs, we evaluate the proposed model in the following classification scenario. Only *Central* entity-document pairs are treated as positive instances, and the remainders as negative instances. We denote this scenario as *Central Only* in the following descriptions. We note that the *Central Only* scenario is the main task of KBA-CCR-2012.

6.3 Evaluation metrics

Since we train a global model regardless of specific entity information, we adopt precision, recall and F1 (harmonic mean between precision and recall) as the evaluation measurements in an entity-insensitive manner. In other words, the metrics are calculated based on the test pool of all entity document pairs irrespective of specific entity using KBAScore[4] script provided by the TREC committee. More specifically, we first scale the relevance scores of entity document pairs in the testing set to [1, 1000]. Then, we compute predicted positive and negative instances for each cutoff with a step to all entities. Subsequently, we compute precision, recall and F1 for each cutoff according to the ground truth. Finally, we choose the highest F1 and corresponding precision and recall as the measurements of the models.

6.4 Experimental setting

Except for the $Sigmoid$ function used in the *Output* layer, we use rectified linear units in other layers as non-linear activation functions. For the hyperparameters listed in Table 1, we set $d_{win} \in \{3, 4, 5\}$ with 128 filters each. We also fixed $batch\_size = 64$. The

$$\gamma \in \{1.0, 1.5, 2.0, 2.5, 3.0, 5.0, 7.0, 10.0\}$$

and

$$p \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$$

were directly chosen via a grid search. Training is done through stochastic gradient descent over shuffled mini-batches with Adam algorithm with learning rate $le-3$ using *TesnorFlow*[5] platform.

6.5 Pre-trained word vectors

After cleaning the dataset by keeping 'raw' texts in Sect. 6.1, we build a dictionary using the most common words in

[4] http://trec-kba.org

[5] https://www.tensorflow.org

Table 4 The top three semantically close words of given five words varying iteration steps

Iter	Tuesday	January	government	people	email
Initialization	Flashbacks	Assistant	Putsch	Military-style	Duraev
	Month-over-month	utm_medium=web	Classic	1481	Shawne
	Votel	Thanks	Dec	Community	'Patriarch
50,000 Steps	Beverage	February	Leader	Don't	E-mail
	Monday	March	Federal	Them	Share
	Rodrigo	November	Window	You	Your
100,000 Steps	Monday	February	Leader	You	E-mail
	Friday	March	Federal	Them	Subscribe
	Saturday	April	Window	Dont	Friend
150,000 Steps	Monday	February	Federal	Them	E-mail
	Friday	March	Authorities	Person	Subscribe
	Wednesday	April	Displaying	Dont	Tweet
200,000 Steps	Monday	February	Federal	Citizens	E-mail
	Wednesday	March	Authorities	Them	Tweet
	Friday	April	Displaying	Americans	Subscribe

the dataset with the length of 150,000. Then we represent the entities and documents as sequence indices with the dictionary, where words outside the dictionary are treated as 'unknown' that is the first word in the dictionary. We trained word vectors on the dataset defined in Sect. 6.1 using the skip-gram model [33]. We run the skip-gram model 5 times with the dimensionality of word embedding (vector) be {64, 128, 200, 250, 300} respectively.

In order to show pre-train word vector results semantically, we demonstrate the top 3 semantically closest words for 5 given words including Tuesday, Januray, government, people and email as shown in Table 4. For instance, the top 3 closest words of Tuesday are flashbacks, month-over-month and votel at the initial, but the top 3 closest words of Tuesday change to Monday, Wednesday and Friday at 200,000 iteration steps.

6.6 Experimental methodology

We conducted extensive comparison experiments. First, we conducted SVM and CosSimilarity classifications using doc2vec representation model in the dataset as our baselines. Then, six variants of DeepJoED models are implemented to evaluate the effectiveness of the proposed deep learning architecture.

- *D_DeepJoED*. It is a variant of DeepJoED with pre-trained vectors from *word2vec*. All words including the unknown one are initialized by pre-trained vectors and then updated during training.
- *S_DeepJoED*. It is a variant of DeepJoED with pre-trained vectors from *word2vec*. All words including the unknown one are initialized by pre-trained vectors, yet kept static, and only other variables of the model are learned.
- *R_DeepJoED*. It is a variant of DeepJoED without pre-trained vectors. All words are randomly initialized and then updated during training.
- *DR_DeepJoED*. A variant of DeepJoED has two sets of vectors. One is pre-trained vectors, and the other is randomly initialized vectors. Each set of vectors is treated as a 'channel' and each filter is applied to both channels. All words are initialized using the pre-trained vectors and random vector respectively. The two sets of vectors are learned during training the network.
- *DS_DeepJoED*. A variant of DeepJoED which has also two channels. All words are initialized using pre-trained vectors twice for two channels. Gradients are backpropagated only through one of the channels, and the other channel are kept static.
- *SR_DeepJoED*. It is a variant of DeepJoED with two channels. All words are initialized using the pre-trained and random vectors corresponding to the two channels. Only random vectors are fine-tuned during training.
- *Dov2Vec_SVM*. It is a SVM model using the distributed representations of the entity-document pairs implemented by the skip-gram model [37] as input with the dimensionality of 128. This model is a baseline.
- *Doc2Vec_CosSim*. It is a simple cosine similarity baseline model. The entities and documents are represented

Table 5 The semantic features between entities and documents

Feature	Description
$N(e_{rel})$	The number of relevant entities occurs in e's profile page
$N(d, e)$	The number of occurrences of e in document d
$N(d, e_{rel})$	The number of e's related entity occurrences in document d
$FPOS(d, e)$	First occurrence position of e in d
$FPOS_n(d, e)$	$FPOS(d, e)$ normalized by the document length
$LPOS(d, e)$	Last occurrence position of e in d
$LPOS_n(d, e)$	$LPOS(d, e)$ normalized by the document length
$Spread(d, e)$	$LPOS(d, e) - FPOS(d, e)$
$Spread_n(d, e)$	$Spread(d, e)$ normalized by document length
$Source(d)$	The source of document d
$weekday(d)$	The weekday of document d published
$burst(d)$	The burst weights of document d

as vectors implemented by the skip-gram model [37] with the dimensionality of 128. The relevance score of an entity-document pair is computed by cosine function.

For reference, we also include three top-ranked approaches on TREC-KBA-2012 dataset as baselines.

- *2-Step J48* [12]. It first detects documents from the stream data that mention the target entity, and then classifies the documents as central or not to the target entity using J48 decision tree classification approach.
- *HLTCOE* [8]. HLTCOE is a Support Vector Machine classifier using bag-of-words and bag-of-entity-names with binary features, representing whether or not a term was present in a document, regardless of its frequency.
- *Random forest* [9]. It is a pointwise learning to rank method implemented by Balog and Ramampiaro and published in SIGIR 2013. It used a rich feature set with 68 various hand-crafted features, and the learning to rank method obtained the best performances compared to many other classification and ranking approaches.

Moreover, we conduct an additional experiment that verifies the performance of embedding features combined with conventional hand-crafted features. The experiment concatenates embedding features that the *D_DeepJoED* model (which is the model with best performance with embedding features only) learns at *Interaction* layer in the Fig. 1 with conventional semantic features between entities and documents in the TREC-KBA-2012 dataset. The combined model is called as *D2HC_DeepJoED*. The selected conventional semantic features, which have been employed effectively in the paper [8], are listed in Table 5.

Table 6 Overall results of evaluated methods

Methods	Central Only		
	Precision	Recall	F1
D_DeepJoED	0.387	0.578	**0.464**
S_DeepJoED	0.356	0.608	0.449
R_DeepJoED	0.369	0.532	0.436
D2HC_DeepJoED	0.386	0.602	**0.470**
DS_DeepJoED	0.332	0.601	0.428
DR_DeepJoED	0.349	0.648	0.454
SR_DeepJoED	0.358	0.556	0.436
Doc2Vec_SVM	0.291	0.616	0.395
Doc2Vec_CosSim	0.171	0.969	0.290
2-step J48	0.243	0.715	0.362
HLTCOE	0.310	0.527	0.391
Random forests	0.369	0.563	0.444

6.7 Results and discussion

The overall results of all the experimental approaches are reported in Table 6. Note that six variants of DeepJoED have the same 128 dimensionality of word embeddings.

In comparison to the baselines listed in the 4th block of Table 6, all variants of DeepJoED achieve higher or competitive F1 scores considerably. Compared with the Dov2Vec_SVM and Doc2Vec_CosSim baselines, our best D_DeepJoED with F1 scores in bold style in 2nd block of Table 6 model improves F1 scores about 18% and 60%, respectively. Moreover, in contrast to the Dov2Vec_SVM and Doc2Vec_CosSim baselines, our low DS_DeepJoED model still increases F1 scores about 8% and 46%, respectively. These results show our DeepJoED model can capture more effective latent semantic features of entities and documents than Doc2Vec which captures latent word level semantic features.

In contrast to other baselines listed in the 5th block of Table 6, which use handcraft features and powerful machine learning models, our DeepJoED model achieves better results in most cases. For 2-step J48 and HLTCOE, all variants of DeepJoED achieve performance gains significantly. The top D_DeepJoED model improve F1 scores about 5% against the top baseline Random Forests. Note that these top baselines leverage various human-designed features that are dedicate to the task of CCR. These results show our DeepJoED model can automatically capture latent semantic features in an end-to-end fashion.

Due to the small size of the dataset, we initially hope that the two channel variants of DeeJoED would prevent over-fitting and thus perform better than single channel variants of DeepJoED. The results are listed in 2nd and

Table 7 Results for different dimensions

Methods	Central Only			
	Dim	Precision	Recall	F1
D_DeepJoED	64	.353	.627	.452
D_DeepJoED	128	.387	.578	**.464**
D_DeepJoED	200	.391	.556	.459
D_DeepJoED	250	.359	.588	.446

3rd blocks of Table 6. From the results, we may conclude that the single channel DeepJoED is better. Moreover, DR_DeepJoED outperforms other two variants DR_DeepJoED and SR_DeepJoED, this shows that the combination of pre-trained vectors of words and random vectors of words is robust for DeepJoED.

For the single channel variants of DeeJoED listed in the 2nd block of Table 6, S_DeepJoED outperforms R_DeepJoED, and D_DeepJoED outperforms S_DeepJoED. These results validate our expectations that (i) the pre-trained vectors are good for DeepJoED, and (ii) the DeepJoED could improve performance gains further when the pre-trained vectors are fine-tuned during training.

In comparison with *D_DeepJoED*, *D2HC_DeepJoED* with F1 scores in bold style in the 2th block of Table 6 achieves marginal performance gains, which uses the same configurations of the deep networks as *D_DeepJoED* except for the conventional semantic features concatenated by the embedding features at the *Interaction* layer in the Fig. 1. This result shows that *D_DeepJoED* have automatically learned the semantic features between entities and documents that already include the hand-crafted features listed in Table 5. Therefore, this result validates that the deep learning approach can learn semantic features between entities and documents in an end-to-end fashion.

6.8 Impact of word vector dimensionality

In this subsection, we show impacts on the DeepJoED by varying dimension of word vectors. We select D_DeepJoED as our reference model, and vary the dimensionality of word vectors while the other settings of D_DeepJoED keeps the same. The results with different dimensionality of word vectors are listed in Table 7. From the results, it can be found that there are no obvious differences in F1 scores when varying the dimensionality of word vectors from 64 to 250. We may conclude that 128 dimension of word vectors is enough for CCR on the TREC-KBA-2012.

7 Conclusion

In this paper, we present a Joint Deep Neural Network Model of Entities and Documents for CCR, termed as DeepJoED. DeepJoED consists of six layers of neural networks. The first four layers are two parallel neural networks sharing the weights and bias, in which one network focuses on learning entity latent semantic representations, and the other learns document latent semantic representations. The fifth layer is an interaction layer to couple the two networks together. The last layer outputs the relevance score of an entity-document pair. DeepJoED takes raw texts of entities and documents as input, and it is learn through stochastic gradient descent over shuffled mini-batches with Adam algorithm. This model has a key point that it learns the distributed representations of entity-document pairs automatically in an end-to-end fashion, so the model can perform on-line learning.

An extensive set of experiments have been conducted on the TREC-KBA-2012 dataset. The results show the model outperforms the state-of-the-art results on the TREC-KBA-2012. The results validate that DeepJoED can learn distributed representation of entities and documents in parallel, and interact with each other as well as effectively detect central documents for a given target entity in KBs.

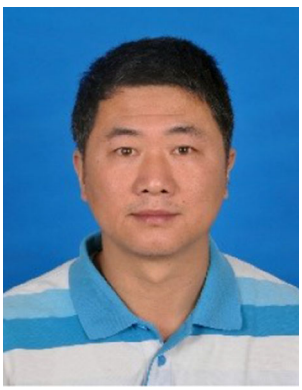
For our future work, we will explore more effective deep learning models such as dependency based neural networks using CNN and RNN, and incorporate the burst feature of entities into the model to improve performance gains for CCR.

Acknowledgements This work has been supported by the National Key Research and Development Program of China under grant (Grant Nos. 2016YFB1000902), the National Natural Science Foundation of China (Grant Nos. 61472040), and the Natural Science Basic Research Plan in Shaanxi Province of China (Grant Nos. 2016JM6082).

References

1. Mihalcea, R., Csomai, A.: Wikify!: linking documents to encyclopedic knowledge. In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pp. 233–242. ACM (2007)
2. Dalton, J., Dietz, L., Allan, J.: Entity query feature expansion using knowledge base links. In: Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, pp. 365–374, ACM (2014)
3. Zhang, C., Zhou, M., Han, X., Zheng, H., Ji, Yang: Knowledge graph embedding for hyper-relational data. Tsinghua Sci. Technol. **22**(02), 185–197 (2017)
4. Dang, H.T., Kelly, D., Lin, J.J.: Overview of the trec 2007 question answering track. In: TREC, vol. 7, pp. 63 (2007)
5. Balog, K., Serdyukov, P., de Vries, A.P.: Overview of the trec 2010 entity track. Technical report, DTIC Document (2010)
6. Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.Y.: Collaborative knowledge base embedding for recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, pp. 353–362, New York, NY, USA (2016)
7. Frank, J.R, Kleiman-Weiner, M., Roberts, D.A., Niu, F., Zhang, C, Ré, C., Soboroff, I.: Building an entity-centric stream filtering test collection for trec 2012. Technical report, DTIC Document (2012)

8. Kjersten, B., McNamee, P.: The hltcoe approach to the trec 2012 kba track. In: TREC. NIST (2012)
9. Balog, K., Ramampiaro, H.: Cumulative citation recommendation: Classification vs. ranking. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pp. 941–944. ACM (2013)
10. Ma, L., Song, D., Liao, L., Wang, J.: Psvm: a preference-enhanced svm model using preference data for classification. Sci. China Inf. Sci. **60**(12), 122103 (2017)
11. Berendsen, R., Meij, E., Odijk, D., de Rijke, M., Weerkamp, W.: The university of amsterdam at trec 2012. In: TREC. NIST (2012)
12. Balog, K., Ramampiaro, H., Takhirov, N., Nørvåg, K.: Multi-step classification approaches to cumulative citation recommendation. In: OAIR, pp. 121–128. ACM (2013)
13. Bonnefoy, L., Bouvier, V., Bellot, P.: A weakly-supervised detection of entity central documents in a stream. In: SIGIR, pp. 769–772. ACM (2013)
14. Wang, J., Song, D., Lin, C.Y., Liao, L.: Bit and msra at trec kba ccr track 2013. In: TREC. NIST (2013)
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
16. Yuan, Z., Yongqiang, L., Xue, Y.: Droiddetector: android malware characterization and detection using deep learning. Tsinghua Sci. Technol. **21**(1), 114–123 (2016)
17. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
18. Greff, K., Srivastava, R.K, Koutník, J., Steunebrink, B.R., Schmidhuber, J.: Lstm: A search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. (2016)
19. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. CoRR, arXiv:1409.2329 (2014)
20. Wen, T.H., Gasic, M., Mrksic, N., Su, P.H., Vandyke, D., Young, S.J.: Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17–21, 2015, pp. 1711–1721 (2015)
21. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2493–2537 (2011)
22. Kim, Y.: Convolutional neural networks for sentence classification. In: Empirical methods in natural language processing, pp. 1746–1751 (2014)
23. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22–27, 2014, Baltimore, MD, USA, Volume 1: Long Papers, pp. 655–665 (2014)
24. Johnson, R., Zhang, T.: Deep pyramid convolutional neural networks for text categorization. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, pp. 562–570 (2017)
25. Shen, Y., He, X., Gao, J., Deng, L., Mesnil, G.: Learning semantic representations using convolutional neural networks for web search. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 373–374. ACM (2014)
26. Shen, Y., He, X., Gao, J., Deng, L., Mesnil, G.: A latent semantic model with convolutional-pooling structure for information retrieval. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM '14, pp. 101–110, New York, NY, USA (2014)
27. Qu, W., Wang, D., Feng, S., Zhang, Y., Yu, G.: A novel cross-modal hashing algorithm based on multimodal deep learning. Sci. China Inf. Sci. **60**(9), 092104 (2017)
28. Zhang, R., Lee, H., Radev, D.R.: Dependency sensitive convolutional neural networks for modeling sentences and documents. In: NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12–17, 2016, pp. 1512–1521 (2016)
29. Lee, J.Y., Dernoncourt, F.: Sequential short-text classification with recurrent and convolutional neural networks. In: NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12–17, 2016, pp. 515–520 (2016)
30. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems, vol. 14, pp. 841–848. MIT Press, Cambridge (2002)
31. Genkin, A., Lewis, D.D., Madigan, D.: Large-scale bayesian logistic regression. Technometrics **49**(3), 291–304 (2007)
32. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: SIGIR, pp. 42–49. ACM (1999)
33. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States., pp. 3111–3119 (2013)
34. Hara, K., Saitoh, D., Shouno, H.: Analysis of function of rectified linear unit used in deep learning. In: 2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12–17, 2015, pp. 1–8 (2015)
35. Zhang, T.: Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, pp. 116, New York, NY, USA (2004)
36. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Improving neural networks by preventing co-adaptation of feature detectors. CoRR, arXiv:1207.0580, (2012)
37. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML, vol. 14, pp. 1188–1196 (2014)



Lerong Ma received a M.S. degree from the Department of Mathematics, Yunnan University, Kunming, China, in 2004. He is currently an associated professor in the School of Mathematics and Computer Science, Yan'an University, Yan'an, China and a PHD student in Beijing Institute of Technology. His research interests include information retrieval, data mining and natural language processing.

Dandan Song received a B.E. degree and Ph.D. degree from the Department of Computer Science and Technology, Tshinghua University, Beijing, China, in 2004 and 2009, respectively. She is currently an associated professor in the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. Her research interests include information retrieval, data mining and bioinformatics.

Lejian Liao received a Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences. He is currently a professor in the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. With main reasearch interest in machine learning, natrual language processing and intelligent network, Professor Liao has published numerous papers in several areas of computer science.

Yao Ni received a B.E. degree from the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, in 2016. He is currently a master candidate in the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include data mining and computer version. He is currently focused on generative adversarial networks.