Neurocomputing 391 (2020) 65–73

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Tag recommendation by text classification with attention-based capsule network

Kai Lei[a,c], Qiuai Fu[a,c], Min Yang[b], Yuzhi Liang[a,*]

[a] Shenzhen Key Lab for Information Centric Networking and Blockchain Technology (ICNLAB), School of Electronic and Computer Engineering (SECE), Peking University, Shenzhen 518055, PR China
[b] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, PR China
[c] PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China

ARTICLE INFO

Article history:
Received 12 October 2018
Revised 13 August 2019
Accepted 24 January 2020
Available online 28 January 2020

Communicated by Dr. Jiliang Tang

Keywords:
Recommendation system
Deep neural networks
Attention model
Capsule network
Routing-by-agreement

ABSTRACT

Tag recommendation has been attracting much attention with the growth of digital resources. The goal of a tag recommendation system is to provide a set of tags for a piece of text to ease the tagging process done manually by a user. These tags have been shown to enhance the capabilities of search engines for navigating, organizing and searching content. However, tag text manually is time-consuming and labor-intensive. In this paper, we introduce a tag recommendation by text classification. We explore the capsule network with dynamic routing for the tag recommendation task. The capsule network encodes the intrinsic spatial relationship between a part and a whole constituting viewpoint invariant knowledge that automatically generalizes to novel viewpoints. In addition, an attention mechanism is incorporated into the capsule network to distill important information from the input documents. We conduct extensive experiments on large publication datasets to evaluate the effectiveness of our model. The experimental results demonstrate that our model substantially outperforms the compared baseline methods and achieves the state-of-the-art results.

1. Introduction

Recommendation system is an efficient approach to overcome the problem of information overload. Tag recommendation allow users to annotate the web pages, musics and articles with key words, and is helpful for searching multimedia content. For example, Last.fm[1] adopts tags to categorize artists and musics; Flickr[2] uses tags to label and organize photos; and CiteULike[3] allows users to tag articles. However, manually tagging is usually time-consuming and labor-intensive. Automatically recommending the appropriate tags becomes a necessary task, which has been widely studied by researchers from different communities.

Generally, existing tag recommendation methods can be divided into personalized collaborative filtering method and object-centered content-based method [1]. The key insight of collaborative filtering methods is to employ the users' historical behavior to recommend subjective and personalized tags, often ignoring the content of the uploaded items [2–4]. One disadvantage of the collaborative filtering methods is the cold-start problem since most items uploaded to the tagging platforms like CiteULike are tagged only by few users. In contrast, the content-based methods is to extract the keywords from the content of the input text and recommend objective tags [5]. The content-based methods can avoid the problem of sparse data without considering users' historical information. Content-based tag recommendation can be realized by keyword extraction [6], topic modeling [7], text classification [8], etc. In this study, we focus on content-based tag recommendation by text classification.

Content-based tag recommendation can be defined as the process of assigning short textual descriptions or key-words (called tags) to a piece of text, which can be modeled as a multi-label text classification task.The current tag recommendation methods are primarily based on deep learning models such as CNN [8] and RNN [9]. However, deep neural network based tag recommendation methods are not accurate enough when the text contents are similar in format. In this case, deep neural networks that classify entities by comparing the "invariance" of the entity and the training samples do not distinguish articles of different topics well. Recently, the capsule network was proposed by Sabour et al. [10] to

* Corresponding author.
E-mail addresses: leik@pkusz.edu.cn (K. Lei), fuqiuai@pku.edu.cn (Q. Fu), min.yang@siat.ac.cn (M. Yang), liangyz@pkusz.edu.cn (Y. Liang).

[1] http://www.lastfm.com.
[2] http://www.flickr.com.
[3] http://www.citeulike.org.

https://doi.org/10.1016/j.neucom.2020.01.091
0925-2312/

address the limitations of deep neural networks. The capsule network encodes the intrinsic spatial relationship between the part and the whole, and classifies the entity by comparing the "equivalent" of the entity with the data in the training set. Inspired by the recent success of the attention mechanism in neural machine translation [11] and speech recognition [12], in this study, we incorporate an attention mechanism into the capsule network to better distill the information from the input text. Experiments results using real world datasets demonstrated the effectiveness of our proposed model.

The primary contributions of this study can be summarized as follows:

- To the best of our knowledge, this is the first work that investigates the capsule networks for tag recommendation.
- We incorporate the attention mechanism into the capsule network to capture the most important textual information in the input documents.
- To evaluate the effectiveness of our model, we conduct extensive experiments on two real-life datasets: TPA and AG. Experimental results shows that our proposed model outperforms the baseline methods on several metrics.

The rest of this paper is organized as follows. In Section 2, we make a brief review of the related work. Section 3 presents the details of our proposed model. In Section 4, we set up the experiments, introducing the experimental data, the evaluation metric, the implementation details, and the baseline methods. The experimental results and analysis are provided in Section 5. Section 6 concludes this manuscript and indicates future work.

2. Related work

In this section, we give a brief overview of the related work, which is divided into three categories: tag recommendation approaches, attention mechanisms, and capsule networks.

2.1. Tag recommendation

Tagging is a widely used mechanism in recommender systems. Various tagging methods have been proposed from different perspectives [13]. We roughly divide existing tag recommendation methods into collaborative filtering method and content-based method.

The key idea of collaborative filtering method is to employ the historical rating information. Feng and Wang [2] modeled a social tagging system as a multi-type graph and recommended tags by learning the weights of nodes and edges in the graph. Fang et al. [4] proposed a novel method for personalized tag recommendation, which could be considered as a nonlinear extension of canonical decomposition. Zhao et al. [3] modeled the relationships in tagging data as a heterogeneous graph and proposed a ranking algorithmic framework for tag recommendation.

In contrast to collaborative filtering methods, content based method takes the content as input and therefore can be used to recommend tags for new content avoiding the cold-start problem of the collaborative filtering methods. One typical content-based tagging technique is using the generative models. Krestel et al. [14] introduced a kind of Latent Dirichlet Allocation model to elicit a shared topical structure from the collaborative tagging efforts of multiple users for recommending hashtags. Si and Sun [15] proposed a LDA-based method which extends the basic LDA graphic model by adding tags and their links to latent topics. In [16], a topical translation model was proposed for microblog tag suggestion in which tag suggestion is modeled as a translation process from content to tags. Godin et al. [7] designed and implemented a binary classifier based on the Naive Bayes technique, which discriminated between English and non-English language tweets for hashtag recommendation.

On the other hand, there are some researches realize tag recommendation by learning a representation to capture the text content via deep neural network based model, then classifies the representation into different tags. Weston et al. [17] proposed a deep CNN based architecture to learn the semantic embeddings from hashtags. The model represents the words as well as the entire textual posts as embeddings in the intermediate layers of the deepCNN architecture. Gong and Zhang [8] adopted CNNs with an attention mechanism to perform tag recommendation task. An extra channel was added to take trigger words into consideration. Meanwhile, [18] utilized an attention-based LSTM for tag recommendation. On the basis of Gong and Zhang [8] and Li et al. [18], Huang et al. [19] proposed an attention-based memory network, which used memory networks to replace CNN and LSTM. Subsequently, Zhang et al. [9] took both textual and image information into consideration for tag recommendation task. Both CNN and RNN are exploited to extract features from images and texts, respectively.

2.2. Attention mechanism

Attention mechanisms can flexibly focus their "attention" to specific parts of the input, paying much attention to the important and discriminative features [11,20]. Attention model was originally proposed in neural machine translation [11]. The attention networks are also widely used in computer vision [20] and speech recognition [12], where the salient part of an image is automatically detected. Bahdanau et al. [12] proposed an attention-based recurrent sequence generators for speech recognition task, which allowed a RNN to learn alignments between sequences of input frames and output labels. Yang et al. [21] took attention on ranking short answer text and extracted the semantically related parts between question-answer pairs. For recommendation tasks, Wang et al. [22] proposed a dynamic attentive deep neural network for article recommendation by learning human editors' demonstration. Gong and Zhang [8], Li et al. [18] utilized attention based deep neural networks to perform hashtag recommendation task.

2.3. Capsule network

Recently, a novel type of neural network is proposed using the concept of capsules to improve the representational limitations of CNN and RNN. Hinton et al. [23] firstly introduced the concept of "capsules" to address the representational limitations of CNNs and RNNs. Capsules with transformation matrices allowed networks to automatically learn part-whole relationships. Consequently, Sabour et al. [10] proposed capsule networks that replaced the scalar-output feature detectors of CNNs with vector-output capsules and max-pooling with routing-by-agreement. The capsule network has shown its potential by achieving a state-of-the-art result on MNIST data. Unlike max-pooling in CNN, however, Capsule network do not throw away information about the precise position of the entity within the region. For low level capsules, location information is "place-coded" by which capsule is active. Xi et al. [24] further tested out the application of capsule networks on CIFAR data with higher dimensionality. Hinton et al. [25] proposed a new iterative routing procedure between capsule layers based on EM algorithm, which achieves significantly better accuracy on the smallNORB dataset. Zhao et al. [26] and Kim et al. [27] presented an empirical exploration of capsule networks for text classification. To date, no work investigate the performance of capsule network in tag recommendation. This study takes the lead in this topic.

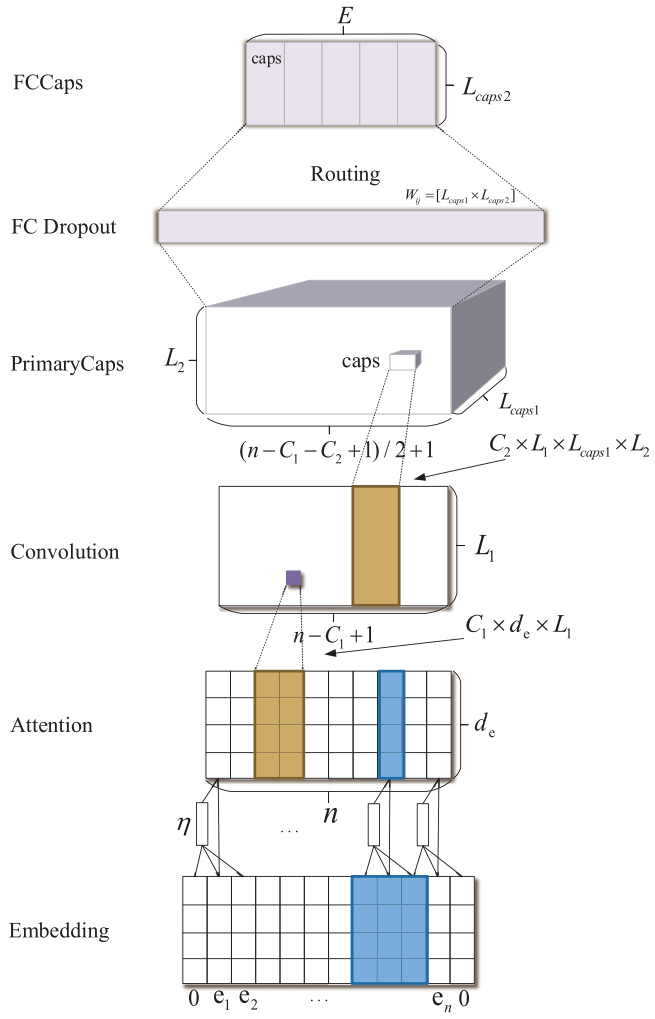

Fig. 1. The architecture of the attention-based Capsule Network(ACN).

3. The proposed model

3.1. Problem definition

Suppose there are a list of training documents $D = \{d_1, d_2 \ldots, d_I\}$, where d_i represents the ith document to be tagged. Moreover, we also have a set of candidate tags $T = \{t_1, t_2 \ldots, t_J\}$, where J is the number of unique tags. Then we can use a document-tag matrix $R = [r_{ij}]_{I\times J}$ to represent the tagging information for all the documents. Here, r_{ij} is a binary variable, where $r_{ij} = 1$ indicates that the tag t_j is associated with document d_i; otherwise, tag t_j is not associated with document d_i. Given an document d_i, the goal of tag recommendation task is to predict the values of $r_i = [r_{i1}, r_{i2} \ldots, r_{ij}]$. Note that, in this paper, we focus on tag recommendation for articles. However, our model is flexible enough to be applied in other applications such as image and video tagging.

3.2. Architecture of our approach

To solve this problem effectively, we propose an attention-based capsule network, as illustrated in Fig. 1. It consists of five layers: an embedding layer, an attention layer, a convolutional layer, a primary capsule layer, and a fully connected capsule layer. The embedding layer converts the words in input text into low-dimensional word embeddings. The attention layer chooses to focus only on a subset of the words of the input text, depending on the attention weights. The convolutional layer learns n-gram features from the text vectors to construct feature maps. And then, the convolutional results are fed into the primary capsule to produce the vector structures. Finally, the fully connected capsule layer generates the tag recommendation results by dynamic routing process based on the output of the primary capsule layer.

3.3. Embedding layer

Suppose $d = \{x_1, x_2 \ldots, x_n\}$ denotes an input document with length n, where x_i denotes the i-th word in document d. Each word x_i is converted into a low-dimensional real-value vector e_i which captures the syntactic and semantic information of the word [28]. Specifically, for each word in d, we first look up the embedding matrix $W^{wrd} \in \mathbb{R}^{d_e\times|V|}$, where d_e is the size of the word embedding and $|V|$ is the size of vocabulary. W^{wrd} is a parameter be to learned and d_e is a hyper-parameter pre-specified by users. Formally, we transform word x_i into its corresponding word embedding e_i by using the matrix-vector product:

$$e_i = W^{wrd}\mu_i \tag{1}$$

where μ_i is one-hot vector of word x_i, which is of size $|V|$. v_i has value 1 at index e_i and 0 in all other positions. Then we can get the real-valued vectors $\mathbf{e} = \{e_1, e_2, \ldots, e_n\} \in \mathbb{R}^{n\times d_e}$ of the input document. Here, we let $\mathbf{e}_{i:i+l}$ refer to the concatenation of l words $\{e_i, e_{i+1}, \ldots, e_{i+l}\}$.

3.4. Attention layer

We incorporate the attention mechanism into capsule network to extract the words that are important and distinguishable for tag recommendation. In theory, the attention layer can be placed before the convolutional layer or after the convolutional layer. We conducted experiments by using different attention designs, and the design used in this manuscript achieves the best results. This may be because that when the attention mechanism is used before the convolutional layer, the attention layer assigns different weights to each word in the input article. On the other hand, if the attention mechanism is used after the convolutional layer, the attention layer will assign different weights to each feature generated by the convolutional layer. If we set the attention layer after the convolutional layer, we will lose the characteristics of the word location information. In order to simulate the way of thinking and understanding of the human brain, the attention mechanism is applied before the convolutional layer in the proposed model.

Given a threshold value η, the goal of the attention mechanism is to select the words whose attention value is greater than the threshold η. Let $\mathbf{k} = \{k_1, k_2, \ldots, k_n\} \in \mathbb{R}^n$ denotes the attention values of the words. We use $\mathbf{m} \in \mathbb{R}^{w\times d_e}$ to denote the parameter matrix, where w is the attention window size, d_e denoting the length of current context vectors. Specifically, at ith step, if w is an even number, we calculate the attention value k_i of ith word as follows:

$$k_i = f(\mathbf{m} \circ \mathbf{e}_{i-w/2:i+w/2} + b) \tag{2}$$

if w is an odd number, k_i is computed by:

$$k_i = f(\mathbf{m} \circ \mathbf{e}_{i-(w-1)/2:i+(w+1)/2} + b) \tag{3}$$

where $\circ$ is element-wise multiplication, f is a non-linear function, b is bias. We use zero-padding to full up the edge points. After getting the attention values for all the words, we compare it to the threshold value η. We extract these words whose attention value is greater than the threshold value as the trigger words, following [8]:

$$\hat{e}_i = \begin{cases} e_i, & k_i > \eta \\ 0, & k_i \le \eta \end{cases} \qquad 1 \le i \le n, \tag{4}$$

where η is the threshold and k_i is attention value of the ith word. Obviously, in order to ensure that those genuine trigger words are selected correctly, the value of η cannot be specified arbitrarily. The choice of the η should depend on the $\mathbf{k}$ which has been calculated previously. We can define η by the following operation:

$$\eta = \rho \cdot \min\{\mathbf{k}\} + (1-\rho) \cdot \max\{\mathbf{k}\} \tag{5}$$

where $\mathbf{k}$ is a sequence of attention values for the words in the input document. $\rho \in [0, 1]$ is a hyper-parameter to balance the minimum and maximum values.

After completing the attention process, we obtained the trigger words $\hat{\mathbf{e}} \in \mathbb{R}^{n \times d_e}$ for the input document, which is the concatenation of words $\{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_n\}$.

3.5. Convolutional layer

This layer is a regular convolutional layer. It extracts the n-gram features of the input sequence at different positions through the convolution operation. Each neuron of the convolutional layer is connected to a local area of the upper layer through a set of weights. The local wighted sum will be passed to a non-linear activate function to produce the final output value of each neuron in the convolutional layer.

Suppose that $\mathbf{w}_i \in \mathbb{R}^{C_1 \times d_e}$ be the ith filter for the convolution operation, where C_1 is the filter size and d_e is the dimension of the input vector. The convolutional layer has L_1 filters with stride of 1. Let $\mathbf{m}_i$ denote the ith feature map of the convolutional layer. Formally, $\mathbf{m}_i$ can be computed as follows:

$$\mathbf{m}_i = f_{cov}(\hat{\mathbf{e}}_{i:i+C_1-1} \otimes \mathbf{w}_i + \mathbf{b}_i) \in \mathbb{R}^{n-C_1+1} \tag{6}$$

where the sign $\otimes$ represents the convolution kernel. $\mathbf{b}_i$ is the bias vector of the ith feature map. And then, we can feed it into the nonlinear activate function f_{cov} and get the results of the ith feature map $\mathbf{m}_i$. Here we use the ReLU function as the activate function. The definition of the ReLU function is as follows:

$$f_{cov}(x) = \max(0, x) \tag{7}$$

We can get the result of the ith feature map through the above process. We have described the process by which one feature is extracted from one filter. Therefore, for $i \in \{1, 2, \ldots, L_1\}$, we can generate L_1 feature maps which can be rearranged as

$$\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{L_1}] \in \mathbb{R}^{(n-C_1+1) \times L_1} \tag{8}$$

3.6. Primary capsule layer

This is the first capsule layer in which the capsules replace the scalar feature detectors of CNNs, which preserves the instantiate parameters such as the position information of words and the grammatical structures of the words.

Suppose that L_{caps1} is the dimension of the primary capsule. Similar to the convolutional layer, we can get the output vector of each capsule by convolution operation. Firstly, we transpose the matrix $\mathbf{M}$ which is the output matrix of the upper layer to get $\mathbf{M}^T = [\mathbf{m}_{\cdot 1}, \mathbf{m}_{\cdot 2}, \ldots, \mathbf{m}_{\cdot (n-C_1+1)}] \in \mathbb{R}^{L_1 \times (n-C_1+1)}$, where $\mathbf{m}_i \in \mathbb{R}^{L_1}$ is the ith column of $\mathbf{M}^T$. Then we define $\mathbf{z}_i \in \mathbb{R}^{C_2 \times L_1}$ as the ith filter for the convolution operation, where C_2 is the filter size and L_1 is the filters of the convolutional layer. A filter $\mathbf{z}_i$ multiplies $\mathbf{m}_i$ one by one with stride of 2 and window size C_2 to produce a $\mathbf{q}_i$ which denotes the ith single feature map of the primary capsule layer. The value $\mathbf{q}_i$ is computed by:

$$\mathbf{q}_i = f_{cov}(\mathbf{m}_{i:i+C_2-1} \otimes \mathbf{z}_i + \mathbf{p}_i) \in \mathbb{R}^{(n-C_1-C_2+1)/2+1} \tag{9}$$

where $\mathbf{p}_i$ is the bias term. Since each capsule includes L_{caps1} neurons, we execute Eq. (9) for L_{caps1} times to get the output vector of the ith capsule as $\mathbf{q}_i' \in R^{((n-C_1-C_2+1)/2+1) \times L_{caps1}}$. Suppose there are L_2 filters. For $i \in \{1, 2, \ldots, L_2\}$, we can obtain the output of the primary capsule layer, which can be computed as

$$\mathbf{Q} = [\mathbf{q}_1', \mathbf{q}_2', \ldots, \mathbf{q}_{L_2}'] \in \mathbb{R}^{((n-C_1-C_2+1)/2+1) \times L_{caps1} \times L_2} \tag{10}$$

3.7. Routing-by-agreement mechanism

Following the strategy in [10], we leverage a non-linear squashing function to convert the length of the input to be [0,1] and preserve the orientation of the input vector. Therefore, the length of the output vectors of capsules can represent the probability that the features are presented in the current capsule layers. The nonlinear squashing function can be viewed as a compression and redistribution way to the input vector which can be computed as

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|} \tag{11}$$

where $\mathbf{v}_j$ is the output vector of jth capsule in the current layer, and $\mathbf{s}_j$ is its total input vector.

The calculation of the vector can be divided into two stages, linear combination and routing process which are denoted as

$$\mathbf{u}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i, \quad \mathbf{s}_j = \sum_i c_{ij}\mathbf{u}_{j|i} \tag{12}$$

where $\mathbf{u}_i$ is the output of ith capsule in the layer below, and $\mathbf{W}_{ij}$ is a weight matrix. $\mathbf{u}_{j|i}$ are the prediction vectors calculated via multiplying the output vector in the layer below by the weight matrix. $\mathbf{c}_{ij}$ are coupling coefficients, which can be determined by the iterative dynamic routing process.

The coupling coefficients between capsule i and all the capsules in the layer above sum to 1 and are determined by a routing leaky-softmax [26] which can be as follows

$$c_{ij} = leaky\text{-}softmax(b_{ij}) \tag{13}$$

where b_{ij} are the logits of coupling coefficients representing the log prior probabilities that capsule i couples to capsule j. We can iteratively update the coupling coefficients $\mathbf{b}_{ij}$ by measuring the consistency between the output vector $\mathbf{v}_j$ of jth capsule in the current layer and the prediction vector $\mathbf{u}_{j|i}$ of ith capsule in the layer below.

$$b_{ij} = b_{ij} + \mathbf{u}_{j|i} \cdot \mathbf{v}_j \tag{14}$$

The routing-by-agreement algorithm is illustrates in Algorithm 1.

Algorithm 1: Routing-by-agreement.

Input: prediction vectors $\mathbf{u}_{j|i}$, current layer l, and routing iteration times r

Output: vector output $\mathbf{v}_j$ of capsule j in l layer

Initialize all the logits of coupling coefficients $b_{ij} \leftarrow 0$

for r *iterations* **do**

  for all capsule i in layer l and capsule j in layer $l+1$: $c_{ij} \leftarrow leaky\text{-}softmax(b_{ij})$;

  for all capsule j in layer $l+1$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\mathbf{u}_{j|i}$, $\mathbf{v}_j \leftarrow squashing(\mathbf{s}_j)$;

  for all capsule i in layer l and capsule j in layer $l+1$: $b_{ij} \leftarrow b_{ij} + \mathbf{u}_{j|i} \cdot \mathbf{v}_j$;

return $\mathbf{v}_j$;

3.8. Fully connected capsule layer

In order to improve the generalization ability of our model, we introduce the dropout mechanism [29] to the network. The

capsule unit of the primary capsule layer will be randomly discarded from the network with a certain probability, and then fully interconnected with each capsule in the fully connected capsule layer. We feed the results after dropout into fully connected capsule layer in which capsules are multiplied by a transformation matrix $\mathbf{W} \in \mathbb{R}^{H \times L_2 \times E}$ followed by routing-by-agreement to produce final capsule $\mathbf{v}_j \in \mathbb{R}^{L_{caps2}}$ and its probability $a_j \in \mathbb{R}$ for each category. Here, H is the number of capsules in the layer below, E is the number of categories, and L_{caps2} is the dimension of the fully connected capsule.

3.9. Objective function

As mentioned above, the coupling coefficients c_{ij} can be iteratively updated by the dynamic routing process. However, the other parameters such as the convolution parameters and the weight matrix need to be learned with the training data. We define the objective function as the margin loss [10] of each category corresponding to each capsule which can be computed by:

$$L_c = \mathbf{I}_c \ \max(0, m^+ - \|\mathbf{v}_c\|)^2 + \lambda(1 - \mathbf{I}_c)\max(0, \|\mathbf{v}_c\| - m^-)^2 \quad (15)$$

where c is the classification category, $\mathbf{I}$ is the indicator function. If the current sample belongs to category c then $\mathbf{I}_c = 1$; otherwise $\mathbf{I}_c = 0$. m^+ is the upper boundary and m^- is the lower boundary which are set as $m^+ = 0.9$ and $m^- = 0.1$. $\|\mathbf{v}_c\|$ is the euclidean distance of the output vector. λ indicates down-weighting of the loss for absent classes, which stops the initial learning from shrinking the lengths of the activity vectors of all the fully connected capsules. The total margin loss L_m is simply the sum of L_c of each fully connected capsule.

In addition, we define the euclidean loss which is based on the idea of the reconstruction loss in image identification domain [10]. First, we let the output of the fully connected capsule layer as the input of the decoder network that consists of three fully connected layers. Thus, we can produce a new output matrix $\mathbf{e}' \in R^{n \times d_e}$. The three fully connected layers contain $n \times d_e/2$, $n \times d_e$ and $n \times d_e$ neurons, respectively. And then, we compute the euclidean distance between each element in the matrix $\mathbf{e}'$ and the corresponding element in the input trigger words $\hat{\mathbf{e}}$. Finally, we calculate the sum of the euclidean distance of each element to get the euclidean loss L_e which can be expressed as

$$L_e = \sum_{i=1}^{n}\sum_{j=1}^{d_e}(\hat{\mathbf{e}}_{ij} - \mathbf{e}'_{ij})^2 \quad (16)$$

where $\hat{\mathbf{e}}_{ij}$ is the element of $\hat{\mathbf{e}}$ and $\mathbf{e}'_{ij}$ is the element of $\mathbf{e}'$. In order to prevent the euclidean loss to manage the entire training process, we reduce the euclidean loss by a certain percentage and then be added to the margin loss to generate the total loss which can be computed as

$$L_t = L_m + \alpha \times L_e \quad (17)$$

where L_t is the total loss, L_m is the margin loss, L_e is the euclidean loss, and α is the scale factor for L_e.

4. Experiments

4.1. Datasets

In order to evaluate the effectiveness of our proposed model, we conduct extensive experiments on two public datasets, i.e., TPA [30] collected from AMiner[4] and AG [31] extracted from ComeToMyHead[5].

[4] http://resource.aminer.org/lab-datasets/crossdomain/.

[5] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html.

Table 1
Statistics of the Datasets.

Dataset	Tag	Item Count
TPA	Database	5059
	Visualization	4074
	Theory	3995
	Medical Informatics	3066
	Data Mining	2270
AG	World	31,900
	Sports	31,900
	Business	31,900
	Sci/Tech	31,900

The TPA dataset contains 18,464 academic articles and 5 types of tags, each with a tag. In our experiment, the "title" and "abstract" parts of an article are used as the text features. We randomly selected 70% of the data in the TPA dataset as the training data, and the reset of the data is used as the test data.

The AG dataset contains 127,600 news articles, belonging to 4 types of tags, each tag contains 30,000 training samples and 1900 test samples. Totally, there are 120,000 training samples and 7600 testing samples in the AG dataset.

In our experiments, we used 30% of the training data as a validation set. We set the model parameters based on the model performance on the validation set.

The detailed information of the datasets are listed in Table 1.

4.2. Evaluation metrics

To evaluate our proposed model effectively, we use precision ($macro-P$), recall ($macro-R$), F_1-score ($macro-F_1$) as the evaluation metrics. We first calculate the precision and recall of each category, which can be computed by:

$$P = \frac{TP}{TP + FP} \quad (18)$$

$$R = \frac{TP}{TP + FN} \quad (19)$$

where TP is the true positive, FP is the false positive, FN is the false negative in the confusion matrix. Then we calculate the average value of precision and recall of each category to obtain the evaluation indicators macro-P, macro-R and macro-F_1, which is formally defined as

$$macro\text{-}P = \frac{1}{n}\sum_{i=1}^{n} P_i \quad (20)$$

$$macro\text{-}R = \frac{1}{n}\sum_{i=1}^{n} R_i \quad (21)$$

$$macro\text{-}F_1 = \frac{2 \times macro\text{-}P \times macro\text{-}R}{macro\text{-}P + macro\text{-}R} \quad (22)$$

where n is number of categories.

4.3. Implementation details

In the experiments, the size of the embedding vector is set to 400. The embedding matrix is pre-trained with the word2vec [32]. We set the mini-batch size to 128 and use Adam optimizer [33] with 1e-3 learning rate to minimize the total loss. The implementation of the experiments is based on the server with the open source framework Tensorflow1.4.0 [34] and python2.7, CUDA8.0.

The GPU used for the experiments is NVIDIA GeForce GTX 1080Ti with 11G memory.

In this experiments, we compare our proposed model with several strong baselines to evaluate the effectiveness of our model. The baseline methods can be mainly divided into four categories: statistical machine learning methods, LSTM and its variant methods, CNN and its variant methods, and capsule based models. We followed the parameter settings in the original paper of the models.

- **AdaBoost** [35]: AdaBoost is an machine learning algorithm to gather several weak classifiers to form a strong classifier.
- **RF** [36]: A probability-optimized random forest (RF) with consistency.
- **GBDT** [37]: GBDT is a gradient boosting decision tree.
- **LSTM** [38]: LSTM is a popular solution to solve long-distance dependency problems. The backward and forward recurrent neural networks (RNN) each have 1000 hidden units.
- **BiLSTM** [39]: BiLSTM was proposed for relation classification. They model the sentence with complete, sequential information about all words. In our experiment, the LSTM layer contains 400 units for each direction, and MLP layer contains 1000 units in BiLSTM method.
- **Att-BiLSTM** [28]: Att-BiLSTM utilizes attention mechanism to capture the most important semantic information in a sentence based on BiLSTM. In this experiment, we combine dropout which is employed on the embedding layer, LSTM layer and the penultimate layer and L2 regularization with Att-BiLSTM to alleviate overfitting.
- **CNN** [40]: CNN is a deep feedforward neural network with convolution operation. The filter windows are set as 3, 4, 5 with 100 feature maps each.
- **ABCNN** [41]: ABCNN is attention-based convolutional neural network, which is originally proposed to handle sentence pair modeling task. We exploit ABCNN-3 which stacks the ABCNN-1 and the ABCNN-2 to perform our experiment.
- **VD-CNN** [42]: VD-CNN is very deep convolutional network, which is a variant method of CNN. VD-CNN operates at the character level and only uses small scale convolution kernel to perform convolution operation.
- **CapsNet**[10]: We use the basic capsule network proposed in [10] to perform our tag recommendation task. In this experiment, we set the routing iteration r as 3 and λ as 0.5.
- **Capsule-B** [26]: Capsule-B is a capsule based text classification model. Capsule-B is composed of three parallel networks with filter windows of 3, 4, 5 in the N-gram convolutional layer, where each network is composed of an embedding layer, followed by a 3-gram convolutional layer with 32 filters and a stride of 1 with ReLU nonlinearity, and regular capsule layers.

5. Experimental results and discussion

5.1. Quantitative evaluation

The experimental results of our proposed model ACN and the compared baseline methods on TPA and AG datasets are summarized in Tables 2 and 3, respectively. From the results, we can observe that ACN out-performs other baselines. For TPA dataset, the proposed ACN improves 1.22% on macro-F_1 score over the baseline models. For AG dataset, the performance of ACN is 0.33% better than the second best model in the baselines. This is the first work use capsule network in tag recommendation. We only used a simple version of the capsule network. It is possible to further improve the model performance by using EM routing or multi-task techniques.

Table 2
Comparisons of our proposed method with baselines on the TPA dataset.

Methods	macro-P	macro-R	macro-F_1
AdaBoost	0.751	0.721	0.731
RF	0.744	0.725	0.732
GBDT	0.811	0.789	0.797
LSTM	0.805	0.797	0.798
BiLSTM	0.815	0.811	0.810
Att-BiLSTM	0.819	0.811	0.812
CNN	0.804	0.798	0.800
ABCNN	0.817	0.813	0.811
VD-CNN	0.813	0.813	0.809
CapsNet	0.820	0.815	0.814
Capsule-B	0.818	0.806	0.810
ACN	**0.829**	**0.825**	**0.824**

Table 3
Comparisons of our proposed method with baselines on the AG dataset.

Methods	macro-P	macro-R	macro-F_1
AdaBoost	0.779	0.780	0.779
RF	0.769	0.769	0.768
GBDT	0.820	0.821	0.820
LSTM	0.861	0.862	0.861
BiLSTM	0.882	0.880	0.881
Att-BiLSTM	0.891	0.890	0.890
CNN	0.914	0.908	0.911
ABCNN	0.917	0.913	0.914
VD-CNN	0.913	0.910	0.912
CapsNet	0.921	0.918	0.920
Capsule-B	**0.926**	0.919	0.917
ACN	0.926	**0.922**	**0.923**

The conventional statistical machine learning methods such as AdaBoost and RF perform poorly since they have poor representation ability. The performance of these methods highly relies on the feature engineering that is time-consuming and labor-intensive. The GBDT method achieves much higher performance than the AdaBoost and RF methods since GBDT builds trees one at a time, where each new tree helps to correct errors made by previously trained tree. With each tree added, the model becomes even more effective.

The deep learning approaches outperform the conventional machine learning approaches (AdaBoost and RF) by a large margin because they reduce the need for feature engineering, one of the most time-consuming parts of machine learning practice. Moreover, the Att-BiLSTM and ABCNN stably exceed the LSTM and CNN methods since they adopt the attention mechanism. These attention-based approaches capture important information from the input text with the supervision of the tag information.

The capsule network takes a further step towards recognizing the existence of the important features and encoding their properties into low-dimensional vectors. This verifies the effectiveness of the capsule network for tag recommendation. ACN performs even better than the standard capsule network and Capsule-B by incorporating the attention mechanism.

5.2. Parameters sensitive analysis

In order to explore the influence of the parameters of the proposed model, we investigate five most essential hyper-parameters which are the routing iteration r, the scale factor α defined in Eq. (17), the lambda value λ defined in Eq. (15), the filter size C_1 and C_2 defined in Eq. 6 and Eq. 9, and the attention window size w defined in Eqs. (2) and (3). Due to the limited space, we only report the parameter analysis results on TPA dataset. Fig. 2 shows the effects of routing iteration, scale factor, lambda value and filter size

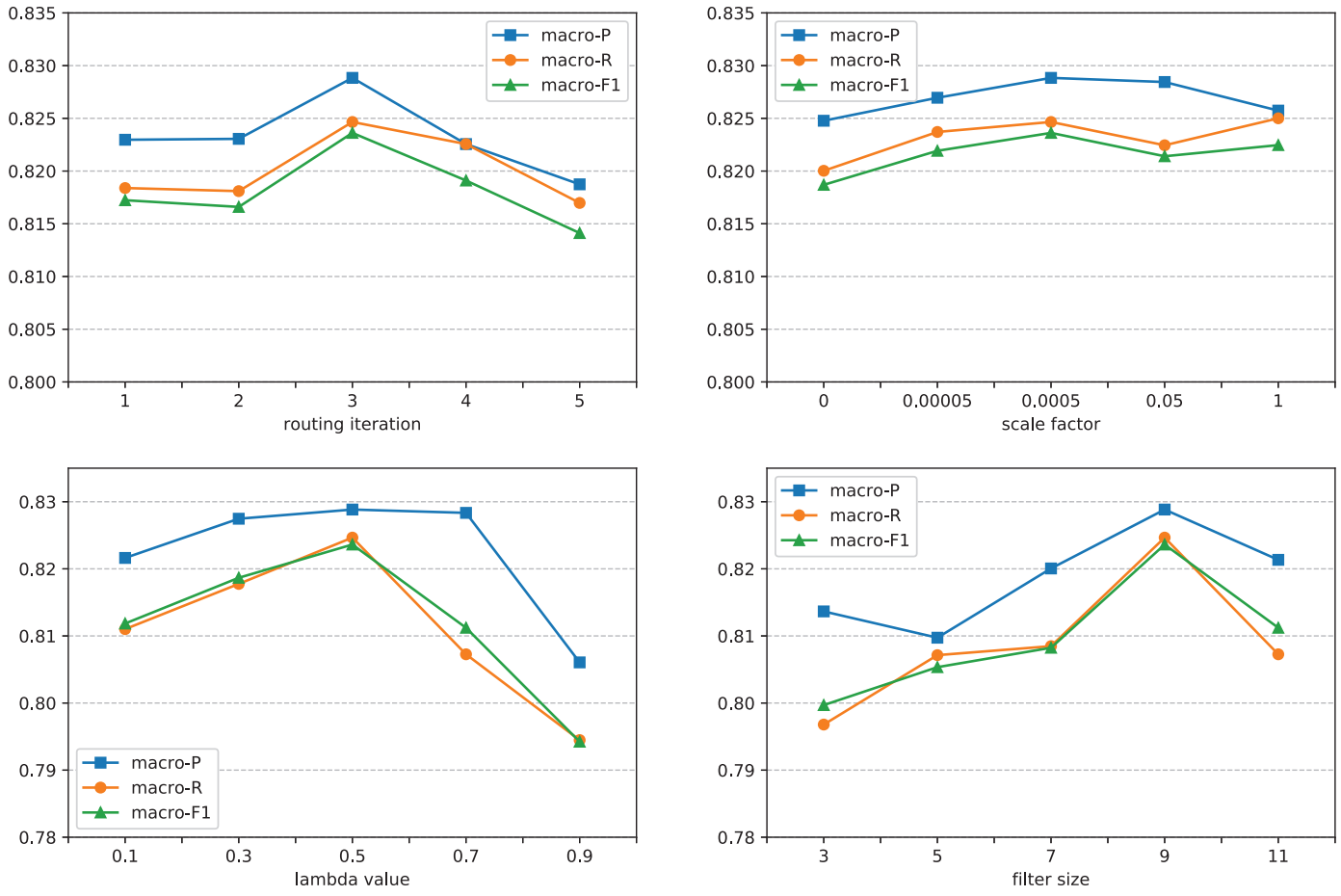

Fig. 2. Exploring the influence of different hyper-parameters on the proposed model.

C_2 on the model. We select five values 1, 2, 3, 4 and 5 of the routing iteration r to test our model. As can be seen from Fig. 2, when the value of the routing iteration is set to 3 we can get the highest macro-P, macro-R and macro-F_1. This may be because that the dynamic routing algorithm is easy to converge.

For the scale factor, we investigate five values [0, 0.00005, 0.0005, 0.05, 1]. From the experimental results, we can see that the value of 0.0005 reaches the highest macro-P and macro-F_1, while the scale factor value of 1 gets the highest macro-R. Taking all factors into consideration, we finally selected 0.0005 as the final scale factor (α) value for our model. The lambda value is the coefficient to balance the margin loss. The selection of its value directly influences the training error of our proposed model. We check the value from [0.1, 0.3, 0.5, 0.7,0.9] to explore its influence on the model. We can see from the experimental results that when the lambda value is set to 0.5, we can get the best results on the constructed evaluation dataset. For the filter size C_2 which connects the convolutional layer and the primary capsule layer, we select the values [3, 5, 7, 9, 11] to investigate its influence to the model. From the experimental results, we can see that the value set to be 9 can obtain the best effects. Similarly, we investigate the influence of different filter size C_1 which connects the attention layer and the convolutional layer.The results show that the models achieve the best results when the filter size is 9.

Table 4 shows the results of our proposed model with different attention window size. "ACN-att-x" indicates the ACN model proposed with the attention window size x. We investigate the window size w as 1, 3, 5, 7 and 9, respectively. Based on the results, we can observe that the performance of ACN gets better with the increase of window size at the beginning, and reaches at the peak when the attention window size w is set to 5. After that, the performance decreases as the window size increases. "ACN-att-1" achieves a bad performance since the importance of the central word merely depends on itself while ignoring the context. The poor performance of "ACN-att-7" and "ACN-att-9" indicates that when the distance of the context words from the central word exceeds a certain range, the dependence of these words with the central word becomes shallower and even produce negative influence.

Table 4
Evaluation results of different attention window size.

Methods	macro-P	macro-R	macro-F_1
ACN-att-1	0.816	0.805	0.807
ACN-att-3	0.823	0.818	0.821
ACN-att-5	0.829	0.825	0.824
ACN-att-7	0.820	0.815	0.816
ACN-att-9	0.813	0.792	0.796

6. Conclusions

In this study, we investigte the capsule network for tag recommendation task. The capsule network can activate higher-level features by agreement between lower-level features. To distill important information from the input sequence, we also incorporate an attention mechanism into the capsule network to capture important and distinguishable features for tag recommendation. We conduct extensive experiments on two large-scale datasets collected

from AMiner and ComeToMyHead, respectively. The experimental results demonstrate that ACN outperforms the compared baseline methods by a large margin in terms of macro-P, macro-R and macro-F_1.

In the future, we will explore the dependency parser to model the long-range contents which are relevant in syntax but far in word order. We also plan to explore capsule network for cross-domain tag recommendation, which is essential when lacking of sufficient labeled training data.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been financially supported by Shenzhen Project (Grant no. ZDSYS201802051831427)

References

[1] F.M. Belém, J.M. Almeida, M.A. Gonçalves, A survey on tag recommendation methods, JASIST 68 (4) (2017) 830–844.
[2] W. Feng, J. Wang, Incorporating heterogeneous information for personalized tag recommendation in social tagging systems, in: KDD, ACM, 2012, pp. 1276–1284.
[3] W. Zhao, Z. Guan, Z. Liu, Ranking on heterogeneous manifolds for tag recommendation in social tagging services, Neurocomputing 148 (2015) 521–534.
[4] X. Fang, R. Pan, G. Cao, X. He, W. Dai, Personalized tag recommendation through nonlinear tensor factorization using gaussian kernel, in: AAAI, AAAI Press, 2015, pp. 439–445.
[5] Y. Wu, S. Xi, Y. Yao, F. Xu, H. Tong, J. Lu, Guiding supervised topic modeling for content based tag recommendation, Neurocomputing 314 (2018) 479–489.
[6] H. Murfi, K. Obermayer, A two-level learning hierarchy of concept based keyword extraction for tag recommendations (2009) 201–214.
[7] F. Godin, V. Slavkovikj, W.D. Neve, B. Schrauwen, R.V. de Walle, Using topic models for twitter hashtag recommendation, in: Proceedings of the International World Wide Web Conferences Steering Committee/ACM, WWW (Companion Volume), 2013, pp. 593–596.
[8] Y. Gong, Q. Zhang, Hashtag recommendation using attention-based convolutional neural network, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2016, pp. 2782–2788.
[9] Q. Zhang, J. Wang, H. Huang, X. Huang, Y. Gong, Hashtag recommendation for multimodal microblog using co-attention network, in: IJCAI, ijcai.org, 2017, pp. 3420–3426.
[10] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, in: NIPS, 2017, pp. 3859–3869.
[11] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate (2014) CoRR abs/1409.0473.
[12] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, Y. Bengio, End-to-end attention-based large vocabulary speech recognition, in: ICASSP, IEEE, 2016, pp. 4945–4949.
[13] Y. Gong, Q. Zhang, X. Huang, Hashtag recommendation for multimodal microblog posts, Neurocomputing 272 (2018) 170–177.
[14] R. Krestel, P. Fankhauser, W. Nejdl, Latent dirichlet allocation for tag recommendation, in: RecSys, ACM, 2009, pp. 61–68.
[15] X. Si, M. Sun, Tag-lda for scalable real-time tag recommendation, J. Comput. Inf. Syst. 6 (1) (2009) 23–31.
[16] Z. Ding, X. Qiu, Q. Zhang, X. Huang, Learning topical translation model for microblog hashtag suggestion, in: IJCAI, IJCAI/AAAI, 2013, pp. 2078–2084.
[17] J. Weston, S. Chopra, K. Adams, #tagspace: Semantic embeddings from hashtags, in: EMNLP, ACL, 2014, pp. 1822–1827.
[18] Y. Li, T. Liu, J. Jiang, L. Zhang, Hashtag recommendation with topical attention-based LSTM, in: COLING, ACL, 2016, pp. 3019–3029.
[19] H. Huang, Q. Zhang, Y. Gong, X. Huang, Hashtag recommendation using end–to-end memory networks with hierarchical attention, in: COLING, ACL, 2016, pp. 943–952.
[20] K. Xu, J. Ba, R. Kiros, K. Cho, A.C. Courville, R. Salakhutdinov, R.S. Zemel, Y. Bengio, Show, attend and tell: Neural image caption generation with visual attention, in: Proceedings of the ICML, volume 37 of JMLR Workshop and Conference, JMLR.org, 2015, pp. 2048–2057.
[21] L. Yang, Q. Ai, J. Guo, W.B. Croft, anmm: Ranking short answer texts with attention-based neural matching model, in: CIKM, ACM, 2016, pp. 287–296.
[22] X. Wang, L. Yu, K. Ren, G. Tao, W. Zhang, Y. Yu, J. Wang, Dynamic attention deep model for article recommendation by learning human editors' demonstration, in: KDD, ACM, 2017, pp. 2051–2059.
[23] G.E. Hinton, A. Krizhevsky, S.D. Wang, Transforming auto-encoders, in: ICANN (1), volume 6791 of Lecture Notes in Computer Science, Springer, 2011, pp. 44–51.
[24] E. Xi, S. Bing, Y. Jin, Capsule network performance on complex data (2017) CoRR abs/1712.03480.
[25] G.E. Hinton, S. Sabour, N. Frosst, Matrix capsules with EM routing, in: International Conference on Learning Representations, 2018 https://openreview.net/forum?id=HJWLfGWRb..
[26] W. Zhao, J. Ye, M. Yang, Z. Lei, S. Zhang, Z. Zhao, Investigating capsule networks with dynamic routing for text classification (2018).
[27] J. Kim, S. Jang, S. Choi, E. Park, Text classification using capsules (2018) CoRR abs/1808.03976.
[28] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, B. Xu, Attention-based bidirectional long short-term memory networks for relation classification, ACL (2), The Association for Computer Linguistics, 2016.
[29] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors (2012) CoRR abs/1207.0580.
[30] J. Tang, S. Wu, J. Sun, H. Su, Cross-domain collaboration recommendation, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012, pp. 1285–1293.
[31] X. Zhang, J.J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: NIPS, 2015, pp. 649–657.
[32] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the International Conference on Neural Information Processing Systems, 2013, pp. 3111–3119.
[33] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, Comput. Sci. (2014).
[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, Tensorflow: large-scale machine learning on heterogeneous distributed systems (2015) CoRR abs/1603.04467.
[35] P. Zhang, Z. Yang, A novel adaboost framework with robust threshold and structural optimization, IEEE Trans. Cybern. 48 (1) (2018) 64–76.
[36] Y. Wang, Q. Tang, S. Xia, J. Wu, X. Zhu, Bernoulli random forests: Closing the gap between theoretical consistency and empirical soundness, in: IJCAI, IJCAI/AAAI Press, 2016, pp. 2167–2173.
[37] J.H. Friedman, Stochastic gradient boosting, Comput. Stat. Data Anal. 38 (4) (2002) 367–378.
[38] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: EMNLP, ACL, 2014, pp. 1724–1734.
[39] S. Zhang, D. Zheng, X. Hu, M. Yang, Bidirectional long short-term memory networks for relation classification, PACLIC, ACL, 2015.
[40] Y. Kim, Convolutional neural networks for sentence classification, in: EMNLP, ACL, 2014, pp. 1746–1751.
[41] W. Yin, H. Schütze, B. Xiang, B. Zhou, ABCNN: attention-based convolutional neural network for modeling sentence pairs, TACL 4 (2016) 259–272.
[42] H. Schwenk, L. Barrault, A. Conneau, Y. LeCun, Very deep convolutional networks for text classification, in: EACL (1), Association for Computational Linguistics, 2017, pp. 1107–1116.

Kai Lei received the B.Sc. degree from Peking University, China, in 1998, the M.Sc. degree from Columbia University in 1999, and the Ph.D. degree from Peking University in 2015, all in computer science. He has worked for companies, including the IBM Thomas J.Watson Research Center, Citigroup, Oracle, and Google from 1999 to 2004. He currently is an Associate Professor with the School of Electronic and Computer Engineering, Peking University, Shenzhen and Director of Shenzhen Key Lab for Informaiton Centric Networking and Blockchain Technologies (ICNLlab.cn). His research interests include data mining, named data networking and blockchain.

Qiuai Fu is a master student in the Department of Computer Science and Technology, Peking University, China. His research interests include natural language processing and recommender system.

Min Yang is currently an assistant professor of Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. She received her Ph.D. degree from the University of Hong Kong in 2017. Her current research interests include machine learning and natural language processing.

Yuzhi Liang is a research assistant in the Department of Computer Science and Technology, Peking University, China. She received her doctor degree from the University of Hong Kong in 2017. Her research interests include artificial intelligence and natural language processing.