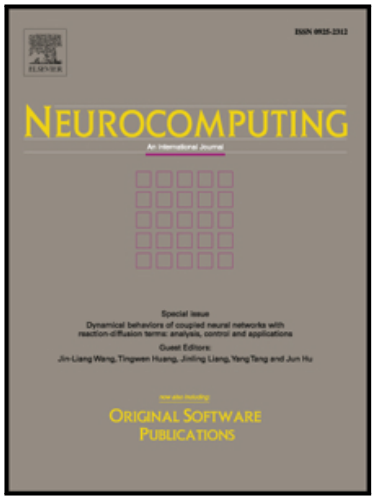
Journal Pre-proof



Keywords Extraction with Deep Neural Network Model

Yu Zhang, Mingxiang Tuo, Qingyu Yin, Le Qi, Xuxiang Wang, Ting Liu

PII: S0925-2312(19)31687-X
DOI: https://doi.org/10.1016/j.neucom.2019.11.083
Reference: NEUCOM 21613

To appear in: *Neurocomputing*

Received date: 30 April 2019
Revised date: 26 August 2019
Accepted date: 26 November 2019

Please cite this article as: Yu Zhang, Mingxiang Tuo, Qingyu Yin, Le Qi, Xuxiang Wang, Ting Liu, Keywords Extraction with Deep Neural Network Model, *Neurocomputing* (2019), doi: https://doi.org/10.1016/j.neucom.2019.11.083

Keywords Extraction with Deep Neural Network Model

Yu Zhang, Mingxiang Tuo, Qingyu Yin, Le Qi, Xuxiang Wang, Ting Liu

Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{*yzhang, mxtuo, qyyin, lqi, xxwang, tliu*}*@ir.hit.edu.cn*

Abstract

Keywords can express the main content of an article or a sentence. Keywords extraction is a critical issue in many Natural Language Processing (NLP) applications and can improve the performance of many NLP systems. The traditional methods of keywords extraction are based on machine learning or graph model. The performance of these methods is influenced by the feature selection and the manually defined rules. In recent years, with the emergence of deep learning technology, learning features automatically with the deep learning algorithm can improve the performance of many tasks. In this paper, we propose a deep neural network model for the task of keywords extraction. We make two extensions on the basis of traditional LSTM model. First, to better utilize both the historic and following contextual information of the given target word, we propose a target center-based LSTM model (TC-LSTM), which learns to encode the target word by considering its contextual information. Second, on the basis of TC-LSTM model, we apply the self-attention mechanism, which enables our model has an ability to focus on informative parts of the associated text. In addition, we also introduce a two-stage training method, which takes advantage of large-scale pseudo training data. Experimental results show the advantage of our method, our model beats all the baseline systems all across the board. And also, the two-stage training method is of great significance for improving the effectiveness of the model.

Keywords: keyword extraction, deep learning, LSTM, attention mechanism, two-stage training.

1. Introduction

Keyword contains the main information that helps people to understand the content of text. By seeing the keywords, it becomes easier for users to determine whether they need the text, which in return will improve the efficiency of Information Retrieval (IR). In addition, due to the refining and conciseness of keywords, people can use them to compute the text correlation with low complexity and therefore, brings benefits to many natural language processing applications like text classification, text clustering, IR and Question Answering system. Admittedly, keywords extraction is a non-trivial task that requires lots of natural language processing techniques. Existing approaches typically apply machine learning algorithm, extracting kinds of features to train their models [1, 2, 3] and achieves great performance.

Although keywords extraction technology has been widely studied, there is still a gap between the best methods and other core Natural Language Processing (NLP) tasks[2]. There exist two main problems that need to be solved for this task. First, traditional approaches highly rely on hand-crafted features, which are labor-intensive. Meanwhile, human-designed features are always incomplete [4], which makes the whole system sensitive to specific datasets. One advantage of deep learning is its ability to enable the computer to automatically learn features directly from input data. Therefore, the issue of these feature-based methods could be alleviated by employing deep neural network models. Second, existing approaches require large-scale annotated corpus to train their model. In other words, in order to get a better model, more datasets are needed [5]. However, it is not easy for human beings to annotate large-scale corpus. Therefore, a desirable solution should be able to generate pseudo training data automatically.

In order to better deal with these problems, in this paper, we propose a novel deep neural network model for the task of keywords extraction. In practice, we regard the whole task as a classification problem. For a given word in the sentence, we train a classifier to determine whether it is a keyword or not. In specifically, to judge the importance of one word, we need to consider the information from two aspects, i.e., the content of the current word and its contextual information. To achieve this goal, we

build our model on the basis of LSTM architecture. Instead of directly employing the traditional LSTM architecture, we propose a target center-based LSTM (TC-LSTM) model. The novel idea behind our TC-LSTM model is to model both the preceding and following contexts of the given word at the same time. By doing so, our model can capture the sentence-level information of the current word and therefore leverage the global information for the given word. By using the historic and the following text information of the target word, our model is able to determine whether it is a keyword or not at the semantic level.

In addition, we also notice that the words in the same sentence contain different information for modeling the specific word. More specifically, some words are more informative than others when representing the given word. For example, in the sentence "Hello everyone, I will introduce a new Chinese book today", the words "book" is more informative than others if we want to determine the importance of the word "Chinese". Therefore, our model should be able to explicitly identify the importance of each word and then generate the representation vector of the given word in a more natural way. Inspired by the recent success of attention mechanism [6, 7, 8, 9], we here employ a self-attention mechanism on top of the TC-LSTM. With the help of the self-attention mechanism, our model is able to generate the representation vector of the given word by considering the importance of the different word and thus brings better results.

Moreover, because the deep learning network models need large-scale training data, and the size of manually annotated training data is very limited, which cannot meet the training requirements of the model. We here propose a new way of generating the training corpus for this task. By utilizing the online search engine, we gain a large-scale inaccurate annotated training corpus. We regard this training corpus a pseudo training data and propose a two-stage training method to better use this dataset. First, we pre-train our model with the inaccurate annotation pseudo training data. Then, we re-train the model by using manually annotated data. In this way, our model gains a much better performance than that using only the manually annotated data.

The main contributions of this paper are three-fold:

- We propose a target-center based LSTM (TC-LSTM) model that helps to gener-

ate the representation vector of a given word by utilizing both its preceding and following context information.

- On purpose of better revealing the importance of different words, we employ the self-attention mechanism on the basis of the TC-LSTM model. With the help of attention mechanism, our method learns to explicitly capture the important words that contain the necessary information for determining keywords.

- We propose a method that can automatically generate large-scale inaccurate annotated training corpus. After that, we employ the two-stage training method that first pre-train the model with these large-scale pseudo training data and then re-train the model by using the manually annotated data.

The rest of this paper is organized as follow: our model and training method for keywords extraction are presented in Section 2 and 3; experiments results and analysis are discussed in Section 4; Section 5 introduces the related work; and the last section concludes the paper and discusses our future work.

2. Related Work

Keywords can represent the core semantics of a piece of text, so how to measure the importance of words in the text is of paramount importance. For the keywords extraction task, many researchers have put forward many valuable research results through continuous exploration. As early as 1958, the computer was introduced into text extraction[10]. Until now, keywords extraction has been developing for more than half a century.

KP-Miner is a kind of method for keywords extraction based on statistics [11]. Firstly, it selects the word sequence separated by punctuation and stop words as candidate keywords. It then filters candidate words by frequency and rules and calculates the overall weight of the candidate phrases, including TF-IDF and the position and length of the phrase two auxiliary features. Finally, a ranking method is employed that rank the weights to extract the keywords and key phrases. This method is simple to implement. However, the rules are always difficult to define and only the word frequency and

other information are far less enough to correctly extract keywords. Therefore, such a method cannot be easily applied to other corpora. On the other hand, for Chinese documents, because there is no obvious word boundary in Chinese text, it affects the quality of keywords extraction to a certain extent and increases the difficulty of Chinese keywords extraction. Chien [12] proposed a character-based Chinese keywords extraction algorithm and constructed a character-level PAT tree obtaining the word string, at the same time using the mutual information for phrase recognition, effectively avoiding the segmentation process.

To better deal with the problem, machine learning-based methods are proposed. For these methods, keywords extraction is regarded as a classification problem, and kinds of features are selected to train the machine learning-based classifier. Features are usually divided into two categories: the internal characteristics of the dataset and the characteristics of the external resources. The internal characteristics of data sets are calculated from the training set, such as TF-IDF, co-occurrence frequency, the location of words for the first time, and so on. The feature based on external resources is calculated from the documents outside the training set, such as keywords based on Wikipedia [3], a search engine's search log [13], the semantic similarity between keywords [14], and so on. Turney [1] proposed C4.5 decision tree and Witten et al. [15] used Bayes model for keywords extraction. In the method of statistical machine learning, the selection of features has a great influence on the effect of keywords extraction.

The graph model-based method for keywords extraction has developed rapidly. Inspired by PageRank [16], Mihalcea and Tarau [17] put forward the TextRank algorithm. Links between words are built based on the semantic relations and using the TextRank algorithm to iterate and extract the keywords. Gollapalli and Caragea [18] proposed a keywords extraction method based on CiteTextRank algorithm [18], and improve the accuracy of keywords extraction by constructing the word network. Zhang et al. [19] proposed a keywords extraction algorithm based on KeyGraph [20], constructing the word network.

In recent years, deep learning methods [21] have been explored widely in many fields [22, 23, 24, 25, 26, 5]. For the task of keywords extraction, researchers propose lots of deep neural network models and achieve great performance. Meng et al. [27]

propose a generative model for keywords prediction with an encoder-decoder framework, which learns to capture the semantics of contexts with the deep neural network model. Zhang and Xiao [28] propose a sequence-to-sequence neural network model with the attention mechanism, copy mechanism, and coverage mechanism. Chen et al. [29] propose a novel sequence-to-sequence architecture that deals with the correlation among different keywords. With the help of correlation constrains, their model can better deal with the duplication and coverage problems and achieve great performance.

3. Methodology

We regard the keywords extraction task as a classification problem. For a given sentence, we first do the word segmentation with the help of LTP [1], which segments the sentence into words. After that, for each of the word in the sentence, we use the classifier to determine whether it is a keyword or not. The flow chart of this process is shown in Figure 1. In practice, we employ a deep neural network to build our classifier, whose inputs are the context words of the current word. For each word, in order to capture its semantic information, we represent each word as a low dimensional, continuous and real-valued vector, also known as word embedding. All the word embeddings are stacked in an embedding matrix $L_w \in \mathbb{R}^{d \times |V|}$, where d is the dimension of word embedding vector and $|V|$ is the size of word vocabulary. The word embedding of w_i is notated as $e_i \in \mathbb{R}^{d \times 1}$, which is the i-th column in the embedding matrix L_w.

Admittedly, deep learning methods can not only effectively avoid the complicated feature engineering, but also take advantage of the pre-trained word embedding vectors to make full use of the semantic information of words, all of which bring benefits to help identify the keywords. Therefore, in this paper, we investigate deep neural network models for this task. We first tried the LSTM-based keywords extraction model and then put forward the target-center based LSTM model. After that, we study the attention-based method. In the following parts of this section, we will introduce these three models in details.

[1] http://www.ltp-cloud.com/

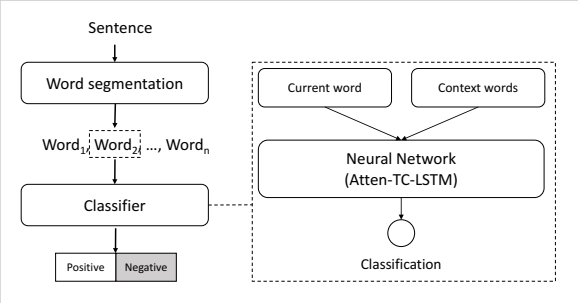


Figure 1: Flow chart of our method.

3.1. LSTM-based Model

Compared with the traditional feed-forward neural network, LSTM can better store the historical information and overcome the shortcoming of modeling the indefinite long sequence inputs. Therefore, LSTM-based models gain better results in many NLP tasks. More specifically, LSTM stores the history information of a sequence of words in the real-valued vector and uses this vector to iteratively link the entire sequence together. In general, this representation vector contains all the history information, we thus employ the LSTM to model the representation vector of the target word. For a given target word, we model the target word and the previous context words into a representation vector by LSTM to predict the probability of the target word to be a keyword.

We build our classifier by using the LSTM, which predicts the probability of being a keyword for each word in the question. For a given target word, we extract a sequence of words from the beginning of the sentence to the target word itself. We then generate the input vectors by utilizing the counterpart embeddings of these words and feed them into the LSTM. We then generate the representation vector of the target word by utilizing the last hidden vector of the LSTM and use this vector to calculate the probability of whether the target word is a keyword or not.

Figure 2 shows the structure of our LSTM-based model, where X_i represents the embedding vector of the i-th word in the sentence, and h_i represents the output of the

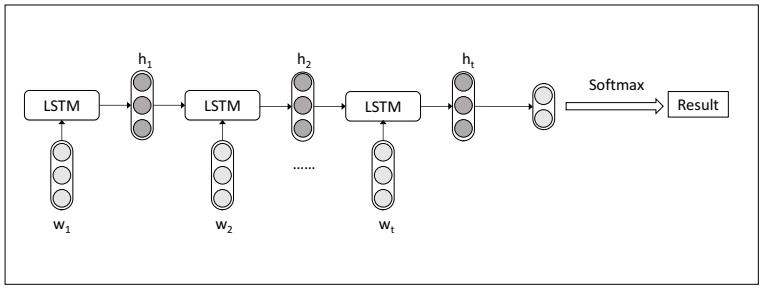


Figure 2: Architecture of the LSTM-based model, where w_i indicates the i-th word in the sentence, and w_t represents the target word.

hidden layer at i-th moment. The operation in each hidden layer can be expressed as:

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \widetilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t * C_{t-1} + i_t * \widetilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{1}$$

It can be abbreviated as:

$$h_i = \mathrm{LSTM}(h_{i-1}) \tag{2}$$

We feed the hidden vector of LSTM at the last step to a linear layer, which outputs a vector with a dimension of 2. By going through the $Softmax$ layer, our model can finally generate the classification of whether the target word is a keyword or not. Formally, suppose X_t is the target word, and the process can be described as:

$$P = \mathrm{softmax}(W_l h_t) \tag{3}$$

Taking the question "宁波有什么特产能在上海世博会占有一席之地? (What specialty in Ningbo can take a place in Shanghai World Expo?)" as an example, we want to determine if "世博会(World Expo)" is a keyword or not. Firstly, for the target word and its historical information, namely "宁波(Ningbo)", "有(has)", "什么(what)", "特产(specialty)", "能(can)", "在(in)", "上海(Shanghai)" and "世博会(World Expo)",

we enter their word embedding vectors to LSTM. After that, our model outputs the classification of whether "世博会(World Expo)" is a keyword.

3.2. Target Center-based LSTM Model

In this subsection, we describe our target center-based LSTM (TC-LSTM) model for keywords extraction. The novel idea behind our method is to utilize the whole contextual information of the given target word. To determine the classification results of the current word, we believe that both the historic and following textual information are equally important. However, in the traditional LSTM model, only the words from the beginning of the sentence to the current could be used, thus ignoring the influence of the following context information of the target word.

Therefore, in order to make full use of the contextual information of the target word in the question, we propose a target center-based LSTM model (TC-LSTM). The main idea is to input both the historic and following text information of the target word into the model and to model the target word together with the information in two directions so as to calculate the probability of the target word and judge it to be a keyword or not.

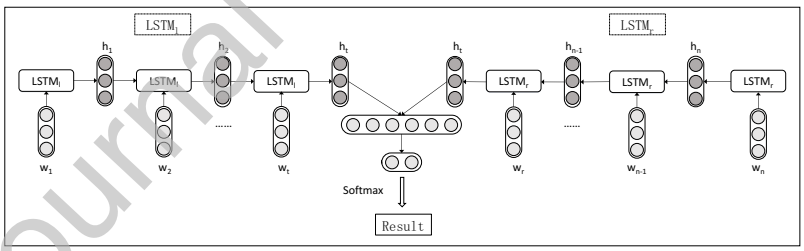


Figure 3: Architecture of the TC-LSTM model, where w_i indicates the i-th word in the sentence (totally n words), and w_t represents the target word.

Figure 3 shows the structure of our TC-LSTM model. Different from the traditional LSTM model, we employ two LSTMs, i.e., the $\mathrm{LSTM_L}$ that models the words from the beginning of the sentence to the current word in a left to right manner, and the $\mathrm{LSTM_R}$ that models the following context information from the end of the sentence to the current word in a reverse direction. Let X_i denote the word vector of the i-th

word of the question, $\overrightarrow{h_i}$ is the output of the hidden layer of $\mathrm{LSTM_L}$ at time i, and $\overleftarrow{h_i}$ represents the output of the hidden layer of $\mathrm{LSTM_R}$ at time i, the operation of hidden layers of $\mathrm{LSTM_L}$ and $\mathrm{LSTM_R}$ can be abbreviated as:

$$\overrightarrow{h_i} = \mathrm{LSTM_L}(\overrightarrow{h_{i-1}}) \tag{4}$$

$$\overleftarrow{h_i} = \mathrm{LSTM_R}(\overleftarrow{h_{i-1}}) \tag{5}$$

The word vector of the target word is always the last input of the LSTM, so that the information of the two directions centered on the target word can be better utilized to represent the importance of the target word. After that, we concatenated the outputs at the last moments of $\mathrm{LSTM_L}$ and $\mathrm{LSTM_R}$, and then feed to the $softmax$ layer to generate the probability distribution of whether the target word is a keyword. Formally, if X_t is the target word, the whole process can be described as follows:

$$P = softmax(W_l(\overrightarrow{h_t} \oplus \overleftarrow{h_t})) \tag{6}$$

where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the outputs of the last hidden layer of $\mathrm{LSTM_L}$ and $\mathrm{LSTM_R}$ respectively, $\oplus$ denotes the concatenation operation.

Take the same sentence as an example. First, for the target word and the counterpart historic information, we feed these word embeddings in the form of vectors, from left to right, to $\mathrm{LSTM_L}$. After that, for the target word and its following context words, namely "世博会(World Expo)", "占有(take)", "一席之地(a place)" and "?", we feed their word vectors from right to left to $\mathrm{LSTM_R}$ in a reverse manner. Finally, our model outputs the probability distribution of whether "世博会(World Expo)" is a keyword.

3.3. Attention-based TC-LSTM Model

Inspired by Lin et al. [8], we investigate the self-attention mechanism for modeling more informative words when predicting the keywords. The basis of our attention-based model is the TC-LSTM proposed in the last subsection, which consists of two standard LSTMs. On top of it, we here apply the attention technique that helps to capture the informative parts of associated contexts.

In practice, we employ the self-attention mechanism that assigns the attention-weight vectors for all the hidden states of the TC-LSTM. After that, the dot option is applied between the attention weights and hidden states to obtain the weighted summation of the representation vector for the current word. For a given sentence that contains n words, $w_1, w_2, ..., w_{target}, ..., w_n$. Suppose we are going to identify whether the target word w_{target} is a keyword, we first generate the representative vector of the preceding and the following text centered by w_{target} by using TC-LSTM architecture:

$$h_t^r = LSTM_r(w_t, h_{t-1}^r) \tag{7}$$

$$h_t^l = LSTM_l(w_t, h_{t-1}^l) \tag{8}$$

where $LSTM_r$ and $LSTM_l$ are two employed LSTMs that model the precedding and following context of the target word independently. We then obtain the hidden vectors for all the words and denote the dimention as u. The hidden vectors of $LSTM_r$ and $LSTM_l$ are representated as $H_r \in \mathbb{R}^{n_r \times u}$ and $H_l \in \mathbb{R}^{n_l \times u}$, seperately:

$$H_r = \{h_1^r, h_2^r, ..., h_{n_r}^r\} \tag{9}$$

$$H_l = \{h_1^l, h_2^l, ..., h_{n_l}^l\} \tag{10}$$

We compute the self-attention weights on the basis of these hidden states. In particular, we calculate the attention-weight matrix A_r (A_l) for H_r (or H_l) as:

$$A_r = softmax(W_2^r tanh(W_1^r H_r^T)) \tag{11}$$

$$A_l = softmax(W_2^l tanh(W_1^l H_l^T)) \tag{12}$$

where W_1 is the parameter with a shape of d_a-by-u and W_2 is in shape of r-by-d_a; r is the number of hops of attention employed. We conduct $softmax()$ along the row vector of each weight matrix. The attention matrix A could be seen as a multi-hope attention matrix, which helps to explicitly capture the informative parts when modeling the current word. Comparing with the traditional single-hop attention mechanism,

multi-hop attention allows our model to focus on different parts of the context and thus effectively capture the sentence-level information from multi-aspects.

The attention matrix A and hidden state vectors H are then multiplied together to generate the weighted summation of the representation vector for the target word:

$$M_r = A_r H_r \tag{13}$$

$$M_r = A_r H_r \tag{14}$$

Finally, we get the representation vector of the current word in both direction by averaging the row vectors in each representative matrix (M_r and M_l), then we concatenate these two vectors to generate the ultimate representation vector of the target word. The rest process of our attention-based model is the same as TC-LSTM, where a $softmax$ layer is applied to generate the probability distribution of whether the target word is a keyword.

4. Two-stage Training Method

It is well known that large-scale training data are required for training the deep neural network models. Nevertheless, for the task of keywords extraction, human efforts are needed to annotate training corpus. Due to the limited number of the annotated dataset, we propose a simple but effective method to automatically annotate keywords in the questions and generate large-scale annotation dataset. However, some of the annotations are inaccurate, thus we call the generated dataset as the pseudo training corpus. To better utilize these datasets, we here present a novel two-stage training method, which pre-train our model by using the inaccurate pseudo training corpus and then retrain our model with manually annotated data. Experimental results have shown that such a method brings better performance. We first present our method for automatically generating the annotation dataset and then describe our two-stage training approach.

4.1. Generating Annotation Data

For the purpose of generating training corpus, we first collect data from websites. In this study, we crawl data from two sources. The first source is the world's largest Chinese interactive QA platform, called "Baidu Zhidao"[2]. For one question in "Baidu Zhidao" website, there not only contains the question itself but also contains all of its description, the best answer, and its similar questions. This information can help us to automatically annotate keywords in the question. Intuitively, if one word is a keyword in a given one question, it will appear for more than one times in the above information, i.e., description, best answer, and similar questions. Our annotation method is built on the basis of this observation.

Another data source we employ is Baidu Open Data[3], logs of user searching results. It records the query entered by the user, as well as the titles and URLs of the web pages the user clicked on. If a user clicks on a page of a question in Baidu Zhidao, it means that the user may need the information in that page, and then the keywords of that question are likely to appear in the query entered by the user.

Therefore, we employ the following steps for automatic keywords annotation. And the flow chart of the process is also shown in Figure 4.

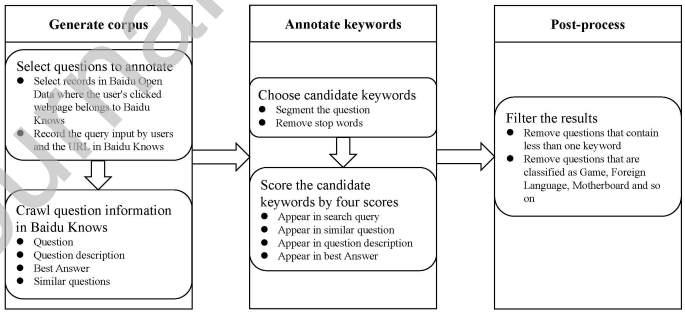


Figure 4: The flow chart of automatic keyword annotation.

(1) Filter questions for annotation. We filter the data in Baidu Open Data and select

[2] zhidao.baidu.com

[3] open.baidu.com

records where the users clicked on webpages that belong to Baidu Zhidao. These records that meet our requirements contain questions that could be annotated, and we also save the counterpart queries the users type in and also the Baidu Zhidao URLs.

(2) Crawl question information. For each record, we crawl the information in the question pages of the filtered URLs by crawlers, including the question content, the description of the question, the best answer to the question and similar questions. All this information can help us to annotate question keywords.

(3) Choose candidate keywords. We segment the question, remove stop words and regard the remaining words as candidate keywords.

(4) Score the candidate keywords. We score each candidate from four aspects, i.e., the query, similar question, question descriptions, and the best answer. Table 4.1 shows the score of each item and the ultimate threshold we choose. For each candidate keyword in the question, if the word appears in the above items, the score of the corresponding item is obtained. We added these scores to get the final score and if the final score exceeds the threshold, the candidate keyword is marked as a keyword. In practice, $\#(sim\_q)$ is the number of similar questions of the question to be annotated, which usually is 4 or 5, $\#(w\_in\_sim\_q)$ is the number of the candidate words that appear in similar questions and their division result is the frequency of occurrence of the candidate word in similar questions.

Items	Score
Search Query	0.75
Similar Questions	$\frac{\#(w\_in\_sim\_q)}{\#(sim\_q)}$
Question Descriptions	0.5
Best Answer	0.25
Threshold	1

Table 1: The score of each item and the threshold.

We believe that words appearing in the search query can better reveal the user's original intent and thus should be more likely to be a keyword. Therefore, we assign this term a high score. For the score of the similar question, if a word appears in sim-

ilar questions frequently, the word is also likely to be the keyword. Hence, we use the frequency of occurrence of the word in similar questions to calculate this score. In addition, the score of question description and best answer are only used as a supplementary reference, so their corresponding scores are lower than others. In addition, we set the score threshold to 1 to filter the false candidates.

(5) Filter the datasets. Finally, we filter the auto-annotated corpus by some rules. Specifically, we remove questions that contain less than one labeled keywords, and that are classified as Game, Foreign Language, Motherboard, Graphics Card, Software, Computer, Camera and Math. After that, we get totally 243882 auto-annotated questions and regard this as the pseudo training data.

4.2. Two-stage Training

It is worthy to note that although we have generated large-scale annotation data to automatically extract keywords from questions, they are sometimes inaccuracy and there is still a gap between these pseudo data and manually annotated data. In order to better utilize these two kinds of the corpus, we here propose a two-stage training approach that can first benefit from the auto-annotated large-scale pseudo corpus and then adapt to real accurate manual annotation dataset.

In the first stage, we pre-train the model by using the auto-annotated large-scale training data and keep the best model. After that, we retrain the whole model in the second stage by utilizing the accurate manual annotated training data. This simple method can substantially improve the performance of our model. For the pseudo training corpus, although we regard it as Inaccurate dataset, most of the keywords, as described in the previous section, can be correctly labeled. Therefore, using the training method that combines large-scale auto-annotated data and accurate manual annotation data is far more effective than the method that just uses manual annotation data alone.

5. Experiments

To illustrate the effectiveness of our methods, we conduct experiments and report results in this section.

5.1. Experimental Setting

As we focus on the keywords extraction task, we first annotate 1000 questions. Among these annotated questions, we select 800 for training and 200 as the test set. As aforementioned, we collect large-scale pseudo training dataset from "Baidu Zhidao", and get 243,882 automatically annotated data for training purpose. Therefore, our dataset consists of three separate parts, i.e., 243,882 auto-annotated dataset for training in the first stage, 800 manual annotated data for the second stage training and 200 for testing. We randomly extract 10% of the training corpus as the validation set. The statics of the dataset is shown in Table 2 All the data will be publicly online once the paper is accepted.

Dataset	Sentence	Training	Testing
Manual Annotated Dataset	1000	800	200
Auto Annotated Dataset	243822	219440	–

Table 2: Statics of the dataset utilized in this paper.

We employ the precision (P), recall (R) and F1-score (F1) as our evaluation matrix, which are defined as:

$$Precision = \frac{\#\ Keywords\ Hit}{\#\ Keywords\ in\ Prediction}$$

$$Recall = \frac{\#\ Keywords\ Hit}{\#\ Keywords\ in\ Gold}$$

where $\#\ Keywords$ presents the number of keywords the model predicted. F1-score is computed as:

$$F = \frac{2 * Recall * Precision}{Recall + Precision}$$

During our experiments, we use the pre-trained word embedding and keep it unchanged during the training process. These word embedding vectors are trained on Sogou news dataset[4] by using the word2vec toolkit [30][5]. We employ the Skip-gram

[4] https://www.sogou.com/labs/resource/ca.php

[5] https://code.google.com/archive/p/word2vec

model to train and the dimension of each word embedding is 100. We set the dimension of each hidden states to 100 and the dimension of d_a utilized to calculate the self-attention is 128. The batch size is set to 128 and $Adam$ (learning rate equals to 0.001) [31] is employed to minimize the training objective. Since we regard the whole task as the classification problem, we utilize cross-entropy as our loss function.

5.2. Experimental Results

We report the experimental results in this subsection. To better evaluate our method, we employ four baseline systems, all of them are based on traditional machine learning and deep learning models.

- MAUI (Multi-purpose Automatic Topic Indexing) The MAUI was proposed in 2010 [32]. MAUI is a machine learning-based approach, which takes the decision tree algorithm to build its classifiers. They extract a set of features, estimate the importance of a target word in multi-aspects, including term frequency, location features, semantic correlation features, specific features (length, POS, naming entity), etc. For all the words in a question, MAUI first selects a set of candidate words and then scores the candidate words. Finally, they select k words with the highest scores as the ultimate keywords.

- TextRank We apply the TextRank [17] algorithm in keywords extraction task. In practice, we rank the importance of words by utilizing the co-occurrence of words. The node of the graph is represented by words. In this way, we can finally identify the keywords from the given sequence.

- MaxEnt (Maximum Entropy Model) MaxEnt is a traditional machine-learning algorithm, which gains great performance in many NLP tasks [33, 34, 35, 36]. We here also build a system based on MaxEnt algorithm. The features used in MaxEnt include the statistical frequency (term frequency, inverse document frequency), morphological features (POS, naming entity), location features, dependency parsing, etc.

- LSTM (Long Short-Term Memory) LSTM is able to store the historical information of a word sequence in a history vector and uses this historical vector to

link the whole sequence together. We believe that such a vector contains historical information that represents the current sequence of words. Meanwhile, this vector contains the necessary semantic information needed from the head of the question to the target word. We here utilize the LSTM to build the baseline system. For this method, the following text information is not used for keywords extraction.

- CopyRNN CopyRNN [29] is the recent deep neural network model that regards the keywords extraction as a generating task. In this method, the model learns to generate the keywords of a given sentence in a sequential manner. We also implement this model in order to make complete comparisons with the exsiting methods.

The results of the experiment are shown in table 3. We can see from the table that,

Methods	P	R	F1
MAUI	61.42	83.08	70.63
TextRank	59.75	58.61	59.17
MaxEnt	78.00	83.09	80.46
LSTM	79.68	86.37	82.89
CopyRNN	76.67	89.64	82.65
TC-LSTM	80.50	86.67	83.47
Att-TC-LSTM (ReLU)$^{\dagger}_{*}$	80.12	87.91	83.83
Att-TC-LSTM$^{\dagger}_{*}$	80.35	88.11	84.05

Table 3: Experimental results on test dataset. $^{\dagger}_{*}$ indicates that our approach is statistical significant over other baselines (using t-test, with $p < 0.05$).

compared with the traditional machine learning-based baselines, the LSTM method can get better results, and the F1 score reaches 82.89%. When we apply the TC-LSTM model, the precision and recall have all been improved, and the F1 score beats the best baseline system – LSTM model by 0.58%. This illustrates the importance of integrating the following context information for keywords extraction. When we

add the following context information of the target word, our model can have a more comprehensive view of the sentence-level information, which in return brings better performance on judging whether the given word is a keyword or not. For example, for the question "谁知道2010世界杯6月11日北京时间几点开幕？(Who knows the opening time of the 2010 World Cup in Beijing on June 11th?)", our TC-LSTM model correctly predicts the "2010" as the keyword while LSTM model cannot. The reason is when we consider only the preceding text information, it is very difficult to predict whether or not "2010" is the keyword and when the following context information is considered, "2010" is predicted as the keyword because we the word "世界杯(World Cup)" is used for the prediction of the keyword.

In addition, we can observe that by applying the self-attention mechanism, our model (Att-TC-LSTM) gain the best performance among all the systems. More specifically, our model with attention get 84.05% in F-score, outperform all the other systems. By utilizing the attention mechanism, our model has an ability to better utilize the informative parts of the given sentence, which in return generates more reasonable representation for the current word. By utilizing this information, our model gains the best performance. Moreover, we also conduct a set of experiments to show the performance of models with different activation function, i.e., replace the $tanh$ utilized in Eq. (11) and (12) by $ReLU$. The results show that the best performance is gained by employing the $tanh$. In sum, all these show the effectiveness of our model proposed in this paper.

However, sometimes when there are no corresponding word vectors for keywords, it is difficult for our models to correctly extract keywords. For example, in the question "2009快乐女声曾轶可个人资料详细介绍？(What is the personal profile of Zeng Yike in the program Super Girl 2009?)". In this sentence, "曾轶可(Zeng Yike)" should be predicted as a keyword, but all our models predict it to be "non-keyword". This is mainly because, in our pre-trained word vectors, there is no word vector for "曾轶可(Zeng Yike)". When we use neural network models to predict the keywords, the embedding for the target word is "None", which brings difficulty for our models to correctly handle the importance of the given word, and as a result, predict the wrong answer. To better handle this problem, one could leverage some hand-crafted features

in our model, which makes our model able to utilize structure information, for instance, the parse information, to predict the keywords.

In order to better verify the effectiveness of the two-stage training method proposed in this paper, we conduct a comparative experiment. In this experiment, we compare the models with the only manual annotated dataset (MAD) and pseudo training dataset (PTD) that trained by using the two-stage training method. As aforementioned, in two-stage training, large-scale data that automatic generation is used for pre-training in the first stage, the manually annotated dataset is then used for training in the second stage based on the pre-training. The results are shown in table 4.

Methods	P	R	F1
LSTM (MAD)	75.18	81.12	78.04
TC-LSTM (MAD)	75.67	81.65	78.54
Att-TC-LSTM (MAD)	74.53	83.44	78.73
LSTM (PTD)	79.68	86.37	82.89
TC-LSTM (PTD)	80.50	86.67	83.47
Att-TC-LSTM (PTD) $^{\dagger}_{*}$	80.35	88.11	84.05

Table 4: The effect of two-stage training for keywords extraction. $^{\dagger}_{*}$ indicates that our approach is statistical significant over other baselines (using t-test, with $p < 0.05$).

We can find that if only the manual annotated dataset is used, the effect of the LSTM model, TC-LSTM model, and Att-TC-LSTM will decrease compared with the ones using the two-stage training method. This can prove that the use of the two-stage training method can improve the overall training effect. This is because that deep neural network models require large-scale training dataset to learn an optimal parameter setting, thus when large-scale training corpus is available, the performance can be increased. However, our large-scale training data are tagged automatically, so there could be some mistakes. In order to correct these errors as much as possible, we retrain our model on the basis of the pre-training in the first stage, and the manually annotated data could be used to adjust parameters, so we can get a better model. The experimental results prove the effectiveness of our two-stage training method.

We also conduct experiments to show the training time of differet methods, i.e., the LSTM, the TC-LSTM and the Att-TC-LSTM models. In particular, we employ the single gpu "GeForce RTX 2080" and implement these models with "CUDA10 + pytorch1.1.0". The results are shown in Table 5. As we can see from this table, our final

Methods	Time (per epoch)
LSTM	27min 06s
TC-LSTM	32min 05s
Attn-TC-LSTM	33min 36s

Table 5: Training time of different models.

model costs about 33 minutes and 36 seconds for each training epoch. Therefore, how to reduce the training costs without the decrement of performance is still challenging.

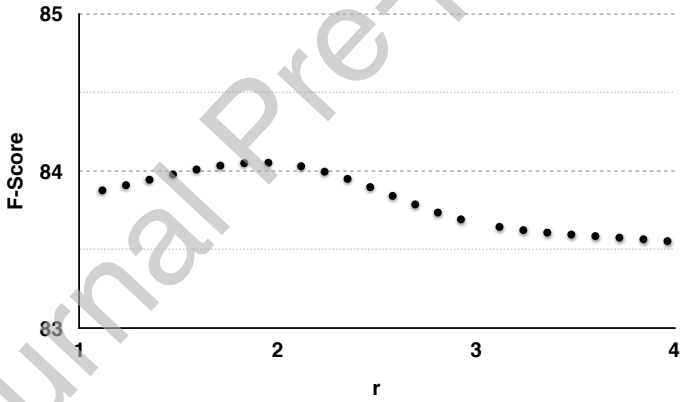


Figure 5: Effect of tuning r for modeling target word.

Lastly, we evaluate the effectiveness of tuning r in the self-attentive mechanism. We vary r from 1 to 4, as is shown in Figure 5. From this picture, we can see that the best performance is gained when $r = 2$. Also, we can find that by improving r from 1 to 2, the performance of our model is increased. This is because that the multi-aspect sentence-level information could bring abundant information when modeling the target word and when r increases, more aspects of information could be captured. Therefore, better performance is gained. However, when r is bigger than 2, the performance

decreased. This is because that in our model, we apply the self-attention mechanism on both the preceding and following context of the current word and $r = 2$ means to focus on totally 4 aspects of the information, which is enough for the length of the question. All these show the benefit of our attention-based model.

6. Conclusion

In this paper, we propose a deep neural network model for the task of keywords extraction. To better utilize the contextual information of a target word, we introduce a target center-based LSTM model. On the basis of this, we apply a self-attention mechanism, which helps our model to captured multi-aspects sentence-level information when determining whether the given word is a keyword or not. In addition, a two-stage training method is proposed, which enable our model to utilize large-scale auto-annotated training corpus. Experiment results reveal that our method improves the effect of keywords extraction. Since the semantic relations between words play an important role for keywords extraction, in our feature work, we will focus on how to make use of the semantic relations between different words and take care of these words that important but with low frequency.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No.61472105).

References

[1] P. D. Turney, Learning to extract keyphrases from text, 2002.

[2] K. S. Hasan, V. Ng, Automatic keyphrase extraction: A survey of the state of the art., in: ACL (1), 2014, pp. 1262–1273.

[3] O. Medelyan, E. Frank, I. H. Witten, Human-competitive tagging using automatic keyphrase extraction, in: Proceedings of the 2009 Conference on Empirical

Methods in Natural Language Processing: Volume 3-Volume 3, Association for Computational Linguistics, 2009, pp. 1318–1327.

[4] J. Schmidhuber, Deep learning in neural networks: An overview, Neural networks 61 (2015) 85–117.

[5] T. Liu, Y. Cui, Q. Yin, W.-N. Zhang, S. Wang, G. Hu, Generating and exploiting large-scale pseudo training data for zero pronoun resolution, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2017, pp. 102–111. URL: http://aclweb.org/anthology/P17-1010. doi:10.18653/v1/P17-1010.

[6] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, Hierarchical attention networks for document classification, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1480–1489.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[8] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, Y. Bengio, A structured self-attentive sentence embedding, arXiv preprint arXiv:1703.03130 (2017).

[9] Q. Yin, Y. Zhang, W. Zhang, T. Liu, W. Y. Wang, Zero pronoun resolution with attention-based neural network, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 13–23.

[10] H. P. Luhn, The automatic creation of literature abstracts, IBM Journal of research and development 2 (1958) 159–165.

[11] S. R. El-Beltagy, Kp-miner: A simple system for effective keyphrase extraction, in: Innovations in Information Technology, 2006, IEEE, 2006, pp. 1–5.

[12] L. F. Chien, Pat-tree-based keyword extraction for chinese information retrieval, Acm Sigir Forum 31 (1997) 50–58.

[13] W.-t. Yih, J. Goodman, V. R. Carvalho, Finding advertising keywords on web pages, in: Proceedings of the 15th international conference on World Wide Web, ACM, 2006, pp. 213–222.

[14] P. D. Turney, Coherent keyphrase extraction via web mining, Proceedings of Ijcai cs.lg/0308033 (2003) 434–439.

[15] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, C. G. Nevill-Manning, Kea: Practical automatic keyphrase extraction, in: Proceedings of the fourth ACM conference on Digital libraries, ACM, 1999, pp. 254–255.

[16] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: Bringing order to the web., Technical Report, Stanford InfoLab, 1999.

[17] R. Mihalcea, P. Tarau, Textrank: Bringing order into text, in: Proceedings of the 2004 conference on empirical methods in natural language processing, 2004, pp. 404–411.

[18] S. D. Gollapalli, C. Caragea, Extracting keyphrases from research papers using citation networks, in: Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.

[19] M. Zhang, H. Geng, X. Wang, Automatic keyword extraction algorithm research using bc method, Journal of Chinese Computer Systems 28 (2007) 189–192.

[20] Y. Ohsawa, N. E. Benson, M. Yachida, Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor, in: Proceedings IEEE International Forum on Research and Technology Advances in Digital Libraries-ADL'98-, IEEE, 1998, pp. 12–18.

[21] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (2015) 436.

[22] C. Hong, J. Yu, X. Chen, Image-based 3d human pose recovery with locality sensitive sparse retrieval, in: 2013 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2013, pp. 2103–2108.

[23] C. Hong, J. Yu, J. Wan, D. Tao, M. Wang, Multimodal deep autoencoder for human pose recovery, IEEE Transactions on Image Processing 24 (2015) 5659–5670.

[24] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, S. Belongie, Kernel pooling for convolutional neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2921–2930.

[25] S. P. Adhikari, C. Yang, K. Slot, M. Strzelecki, H. Kim, Hybrid no-propagation learning for multilayer neural networks, Neurocomputing 321 (2018) 28–35.

[26] A. Liu, Y. Laili, Balance gate controlled deep neural network, Neurocomputing 320 (2018) 183–194.

[27] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, Y. Chi, Deep keyphrase generation, arXiv preprint arXiv:1704.06879 (2017).

[28] Y. Zhang, W. Xiao, Keyphrase generation based on deep seq2seq model, IEEE Access 6 (2018) 46047–46057.

[29] J. Chen, X. Zhang, Y. Wu, Z. Yan, Z. Li, Keyphrase generation with correlation constraints, arXiv preprint arXiv:1808.07185 (2018).

[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.

[31] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[32] O. Medelyan, V. Perrone, I. H. Witten, Subject metadata support powered by maui, in: Proceedings of the 10th annual joint conference on Digital libraries, ACM, 2010, pp. 407–408.

[33] K. Nigam, J. Lafferty, A. McCallum, Using maximum entropy for text classification, in: IJCAI-99 workshop on machine learning for information filtering, volume 1, 1999, pp. 61–67.

[34] M. Osborne, Using maximum entropy for sentence extraction, in: Proceedings of the ACL-02 Workshop on Automatic Summarization-Volume 4, Association for Computational Linguistics, 2002, pp. 1–8.

[35] H. L. Chieu, H. T. Ng, Named entity recognition: a maximum entropy approach using global information, in: Proceedings of the 19th international conference on Computational linguistics-Volume 1, Association for Computational Linguistics, 2002, pp. 1–7.

[36] H. L. Chieu, H. T. Ng, Named entity recognition with a maximum entropy approach, in: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, Association for Computational Linguistics, 2003, pp. 160–163.

Conflict of Interest and Authorship Conformation Form

Please check the following as appropriate:

- o All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.
- o This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue.
- o The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript
- o The following authors have affiliations with organizations with direct or indirect financial interest in the subject matter discussed in the manuscript:

Author's name	Affiliation
Yu Zhang	Harbin Institute of Technology
Mingxiang Tuo	Harbin Institute of Technology
Qingyu Yin	Harbin Institute of Technology
Le Qi	Harbin Institute of Technology
Xuxiang Wang	Harbin Institute of Technology
Ting Liu	Harbin Institute of Technology

Biography of the authors

Yu Zhang is a professor in the Research Center for Social Computing and Information Retrieval, School of Computer Science and Technology, Harbin Institute of Technology. His primary re- search interest is question answering and personalized information retrieval.

Mingxiang Tuo is a master student in the Research Center for Social Computing and Information Retrieval, School of Computer Science and Technology, Harbin Institute of Technology. His research interests lie in natural language processing and question answering.

Qingyu Yin is a Ph.D. student in the Research Center for Social Computing and Information Retrieval, School of Computer Science and Technology, Harbin Institute of Technology. His research interests lie in natural language processing and human-computer dialogue.

Le Qi is a Ph.D. student in the Research Center for Social Computing and Information Retrieval, School of Computer Science and Technology, Harbin Institute of Technology. His research interests lie in natural language processing and question answering system.

Xuxiang Wang is a master student in the Research Center for Social Computing and Information Retrieval, School of Computer Science and Technology, Harbin Institute of Technology. Her research interests lie in natural language processing.

Ting Liu is a professor in the Research Center for Social Computing and Information Retrieval, School of Computer Science and Technology, Harbin Institute of Technology. His primary re- search interest is natural language processing, information retrieval, and social computing.

Pictures

Yu Zhang

Mingxiang Tuo

Qingyu Yin

Le Qi

Xuxiang Wang

Ting Liu