Multi-level Memory Network with CRFs for Keyphrase Extraction

Tao Zhou, Yuxiang Zhang^(✉), and Haoxiang Zhu

College of Computer Science and Technology, Civil Aviation University of China,
Tianjin, China

{zhou.tao1,zhu.hx}@outlook.com, yxzhang@cauc.edu.cn

Abstract. Keyphrase, that concisely describe the high-level topics discussed in a document, are very useful for a wide range of natural language processing (NLP) tasks. Current popular supervised methods for keyphrase extraction commonly cannot effectively utilize the long-range contextual information in text. In this paper, we focus on how to effectively exploit the long-range contextual information to improve the keyphrase extraction performance. Specifically, we propose a multi-level memory network with the conditional random fields (CRFs), which allows to have unrestricted access to the long-range and local contextual information in text. We first design the multi-level memory network with sentence level and document level to enhance the text representation. Then, we integrate the multi-level memory network with the CRFs, which has an advantage in modeling the local contextual information. Compared with the recent state-of-the-art methods, our model can achieve better results through experiments on two datasets.

Keywords: Keyphrase extraction · Sequence labeling · Memory network

1 Introduction

Automatic keyphrase extraction is to recommend a set of representative phrases that are related to the main topics discussed in a document. Since keyphrases can provide a high-level topic description of a document, they are beneficial for a wide range of natural language processing tasks such as information extraction, text summarization and question answering [21]. However, the performance of existing methods is still far from being satisfactory [10]. The main reason is that it is very challenging to determine whether a phrase or sets of phrases can accurately capture main topics that are presented in the document.

Existing methods for keyphrase extraction can be broadly divided into unsupervised and supervised methods. Specifically, unsupervised approaches directly treat keyphrase extraction as a ranking problem, scoring each word using various measures such as TF-IDF (term frequency-inverse document frequency) and graph-based ranking scores (*e.g.*, degree centrality or PageRank score) [7, 8, 22, 27].

Supervised methods usually treat the keyphrase extraction as a binary classification task, in which a classifier is trained on the features of labeled keyphrases to determine whether a candidate phrase is a keyphrase [9, 10, 19]. Compared with unsupervised methods, supervised approaches can yield good results given sufficient training samples.

Recently, EK-CRF [9] employed the CRFs to extract keyphrases from scientific research articles, which was trained on token-based features incorporating linguistic, document structure information, and expert knowledge. This work achieved better performance on keyphrase extraction task and was shown to be state-of-the-art in previous traditional supervised methods. However, if we can not consider the features used in EK-CRF, CRFs only capture local structural dependencies. In addition, EK-CRF mainly relies on manual feature engineering, which may require considerable effort and domain-specific knowledge.

In this work, we aim to capture the long-range contextual information hidden in the text sequence and remove the need of manual feature engineering to extract keyphrases from scientific papers. Specifically, we formulate the keyphrase extraction as a sequence labeling task. We first use the memory network [23] to capture the long-range contextual information hidden in text data. Note that although plain recurrent neural networks (RNNs) can encode the sequential text and their variants such as long short-term memory (LSTM) models can further capture non-local patterns, they still exhibit a significant local bias in practice [14]. In order to make full use of the effective information hidden in text sequence, we extend the input memory of the memory network with two different levels: sentence level and document level. Secondly, we use the CRF model to capture the dependencies between adjacent words in text sequence and determine whether a candidate phrase is a keyphrase. Finally, we conduct comprehensive experiments over two publicly available datasets (KDD and WWW) in Computer Science area. Experimental results show that the proposed approach outperforms several state-of-the-art supervised methods.

The remainder of this paper is organized as follows. We firstly summarize related works on keyphrase extraction and memory networks in Sect. 2. Secondly, the proposed model for keyphrase extraction is described in Sect. 3. Then, the datasets, experimental results and discussions are illustrated in Sect. 4. Finally, we conclude this paper in Sect. 5.

2 Related Work

In this section, we firstly review the related works on keyphrase extraction and then summarize the existing works on the memory network.

2.1 Keyphrase Extraction

As mentioned in Sect. 1, existing approaches for keyphrase extraction can be broadly divided into unsupervised and supervised methods. This work is mainly related to supervised methods which have been proven to be effective in the

keyphrase extraction task. Research on supervised methods has focused on two issues: classifier selection and feature design. Current state-of-the-art classifiers typically include Naïve Bayes [3, 24], decision trees [19], CRFs [9], deep recurrent neural networks (RNN) [16], *etc.* The features used to represent an instance can be broadly divided into three categories: statistical features, structural features and syntactic features [10].

Zhang *et al.* [26] is the first to use the CRFs extracting keyphrases, which provides a way to explore the local contextual information in text sequence and traditional features, to identify each candidate word by sequence labeling. Bhaskar *et al.* [2] employed CRFs trained mainly on linguistic features such as part-of-speech (POS), chunking and named-entity tags for keyphrase extraction. Gollapalli *et al.* [9] also utilized CRFs to extract keyphrases from research papers, which was trained on token-based features incorporating linguistic, document-structure information and expert knowledge. CopyRNN [16] is the first to employ the sequence-to-sequence (Seq2Seq) deep learning model to predict keyphrases for documents. Following CopyRNN, a few extensions have been proposed to help better generate keyphrases [4, 28]. In addition, Alzaidy *et al.* [1] integrated the CRF with the bidirectional long short term memory networks (LSTMs) to extract keyphrases from research papers. However, this method didn't capture the long-range contextual dependencies between words in text. We extend the memory network with stronger storage capacity to jointly maintain local structural information provided by RNNs with long-range dependencies in the long text.

2.2 Memory Network

Despite of the success of the RNNs on various text modeling tasks, simple RNNs still exhibit a significant local bias in practice [25]. Memory network [23] enhances the long-term memory capability of deep network by augmenting the internal memory with a series of extra memory components, and provides a general approach for modeling long-range dependencies and making multi-hop reasoning, which has advanced many NLP tasks such as question answering [20] and reading documents [17]. Sukhbaatar *et al.* [20] proposed the end-to-end memory network, which can be trained end-to-end without any intervention. Kumar *et al.* [12] proposed the dynamic memory network, which uses a sentence-level attention mechanism to update its internal memory during multi-hop inference. Miller *et al.* [17] encoded prior knowledge by introducing a key memory structure which stores facts to address to the relevant memory value. Taking inspiration from these works, we design the multi-level memory network with CRFs to extract keyphrases from research papers.

3 Methodology

In this section, we first present the problem definition in Subsect. 3.1. We then explain the overview of our proposed model in Subsect. 3.2, followed by the details of each component of the model from Subsect. 3.3 to 3.4.

3.1 Problem Definition

Keyphrase extraction is formulated as a task of sequence labeling, predicting a label for each word in the input text sequence. More specifically, we denote the input source document as a sequence $\mathbf{x} = \{x_1, x_2, \dots, x_l\}$, where x_t represents t -th input word and l is the length of the sequence. The goal of the model is to predict a sequence of labels $\mathbf{y} = \{y_1, y_2, \dots, y_l\}$, where each label y_t corresponding to the input word x_t represents whether x_t is a keyphrase word or not keyphrase word.

3.2 Model Overview

Figure 1 illustrates the overview of the Multi-Level Memory network with CRFs (MLM-CRF) for keyphrase extraction. This model includes two main parts: the memory layer, capturing long-range dependencies using the deep memory network; and the CRF layer, capturing local dependencies and labeling each word of input text sequence with five different labels (detailed in Subsect. 4.3).

The memory layer can be divided into two parallel modules at different levels: sentence level memory and document level memory. Each module further includes three components similar to the works [15, 20]: (1) the input memory vector $\mathbf{m}_t$, which captures the information from the word embedding layer of the text sequence; (2) the current input embedding $\mathbf{u}_t$, which is the representation of the current word; and (3) the output vector $\mathbf{c}_t$, which is similar to the input memory $\mathbf{m}_t$.

In addition, the output memory representation of the memory layer $\mathbf{o}$, summarizing the long-range semantic and structure information from the input text sequence without distance limitation, is calculated by a weighted sum over the output representations, in which the attention weights are determined by measuring the similarity between the input memory vector and the current input embedding. Finally, the output of the memory layer $\mathbf{o}$ is fed into the CRF layer to predict keyphrases for documents using sequence labelling model CRF. In the remainder of this section, we will present the MLM-CRF in detail.

3.3 Memory Layer

Input Memory Representation. At first, the word embedding look-up table, trained by GloVe [18], is applied to map each word x_t in the text sequence into an embedding vector $\mathbf{x}_t$. Although we can directly use this embedding vector as the input memory representation in the context of the memory network, in order to tackle the drawback of insensitivity to temporal information between memory cells [15], we obtain the input memory representation $\mathbf{m}_t$ by adopting the bidirectional gate recurrent unit (GRU) [5] to encode the word embedding vectors $\mathbf{x}_t$:

$$\vec{\mathbf{m}}_t = \overrightarrow{\text{GRU}}(\mathbf{x}_t, \vec{\mathbf{m}}_{t-1}) \quad (1)$$

$$\overleftarrow{\mathbf{m}}_t = \overleftarrow{\text{GRU}}(\mathbf{x}_t, \overleftarrow{\mathbf{m}}_{t+1}) \quad (2)$$

Fig. 1. An overview of the MLM-CRF model for keyphrase extraction. It is shown only as a single hop.

$$\mathbf{m}_t = \tanh(\overrightarrow{\mathbf{W}}_m \overrightarrow{\mathbf{m}}_t + \overleftarrow{\mathbf{W}}_m \overleftarrow{\mathbf{m}}_t + \mathbf{b}_m) \quad (3)$$

where $\overrightarrow{\mathbf{W}}_m$, $\overleftarrow{\mathbf{W}}_m$ and $\mathbf{b}_m$ are three trainable parameters to adjust the input memory representation $\mathbf{m}_t$.

Current Input Representation. In order to calculate the attention weight of each element in the input memory, we enforce the current input to be in the same space as the input memory. More specifically, we use the obtained $\mathbf{m}_t$ to represent the current input representation $\mathbf{u}_t$, i.e., $\mathbf{u}_t = \mathbf{m}_t$. Note that as illustrated in Fig. 1, the current input $\mathbf{u}_t$ in the sentence level and the document level is set to the same. We will detail it in the subsection Extensions.

Attention Weight Calculation. We compute the attention weight of each element in the input memory by measuring the relevance between the current input $\mathbf{u}_t$ and each element of the input memory $\mathbf{m}_i$ with a softmax function:

$$p_{t,i} = \text{softmax}(\mathbf{u}_t^T \mathbf{m}_i) \quad (4)$$

where $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$.

Output Memory Representation. Similar to the input memory vector $\mathbf{m}_t$, the output vector $\mathbf{c}_i$ is also the contextual representation vector which is used to capture the contextual semantic information of the text sequence. The output vector is also encoded by bidirectional GRU, but with different parameters in the GRUs function and tanh layers of Eq. (1), (2) and (3).

The output vector is used to generate the final output memory of the memory layer $\mathbf{o}_t$, which is the weighted sum over the attention weight and the output vector, as:

$$\mathbf{o}_t = \sum_i p_{t,i} \mathbf{c}_i. \quad (5)$$

The output memory allows the model to have unrestricted access to elements in previous steps as opposed to a single hidden state in RNNs, which will helps the CRF fully utilize the long-range dependencies in the text sequence to better predict keyphrases for documents.

Extensions. Many existing works [6, 17] discussed the influence of the different lengths covered by the attention mechanism in the memory network. Inspired by these works, we explore the different length of the input memory at two different levels: the sentence level and document level.

In the sentence level, the attention mechanism covers just the sentence containing the current word, and the corresponding attention weight is calculated as:

$$p_{t,i}^s = \text{softmax}(\mathbf{u}_t^T \mathbf{m}_i^s) \quad (6)$$

where $i \in [1, n]$ and n is the length of the sentence.

In the document level, the whole document is covered by the attention mechanism and the attention weight as computed as:

$$p_{t,i}^d = \text{softmax}(\mathbf{u}_t^T \mathbf{m}_i^d) \quad (7)$$

where $i \in [1, l]$ and l is the length of the document. The current input $\mathbf{u}_t$ used in both the sentence level and the document level is set to the same.

Then we calculate the final output memory of the memory layer as:

$$\mathbf{o}_t = \lambda \sum_i p_{t,i}^d \mathbf{c}_i^d + (1 - \lambda) \sum_i p_{t,i}^s \mathbf{c}_i^s \quad (8)$$

where λ is a hyper-parameter, which is set to adjust weight from the different levels. Thus we replace Eq. (4) and (5) with Eq. (6), (7) and (8). The output memory of the memory layer can capture the information from not only the local but also the long-distance context by using the two-level output vectors.

We can further extend the model by stacking multiple memory hops for capturing multiple fact from the memory, which stacks hops between the current input $\mathbf{u}_t$ and the k -th hop $\mathbf{o}_t^k$ to be the input to the $(k + 1)$ -th hop:

$$\mathbf{u}_t^{k+1} = \mathbf{u}_t^k + \mathbf{o}_t^k \quad (9)$$

In our model, we simply limit the hop to only 1.

3.4 CRF Layer

In this section, we feed the output of the memory layer into the CRF layer for extracting keyphrases. CRF [13] has been proven to be effective for sequence labeling tasks. We use CRF jointly with the memory network to predict the sequence of labels for the keyphrase extraction task.

Given the input text sequence $\mathbf{x} = \{x_1, x_2, \dots, x_l\}$ and the sequence of output labels $\mathbf{y} = \{y_1, y_2, \dots, y_l\}$, the score is computed as:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{t=0}^l \mathbf{A}_{y_t, y_{t+1}} + \sum_{t=1}^l \mathbf{P}_{t, y_t}, \quad (10)$$

where $\mathbf{P}$ is the linearly transformed matrix from the output matrix of memory network $\mathbf{U}$ such that $\mathbf{P} = \mathbf{U}\mathbf{W}$. The size of $\mathbf{U}$ is $l \times d$, where d is the size of the output vector $\mathbf{u}$. The size of $\mathbf{W}$ is the weight matrix with the size of $d \times k$, where k is the number of labels. Thus, the size of $\mathbf{P}$ is $l \times k$, where $p_{i,j}$ represents the score of the j -th label of the i -th word of the input sequence. In addition, $\mathbf{A}$ is the matrix of transition scores, in which $a_{i,j}$ represents the score of a transition from the label i to label j .

The probability of the label sequence $\mathbf{y}$ can be calculated by the softmax function as follow:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(s(\mathbf{x}, \mathbf{y}))}{\sum_{\tilde{\mathbf{y}} \in \mathbf{Y}_{\mathbf{x}}} \exp(s(\mathbf{x}, \tilde{\mathbf{y}}))}, \quad (11)$$

where $\mathbf{Y}_{\mathbf{x}}$ represents all possible label sequences given an input sequence $\mathbf{x}$. In the training procedure, we maximize the log-probability of the correct label sequence:

$$\mathcal{L} = \log(p(\mathbf{y}|\mathbf{x})) = s(\mathbf{x}, \mathbf{y}) - \log\left(\sum_{\tilde{\mathbf{y}} \in \mathbf{Y}_{\mathbf{x}}} \exp(s(\mathbf{x}, \tilde{\mathbf{y}}))\right). \quad (12)$$

This objective function and its gradients can be efficiently computed by dynamic programming algorithm. In order to find the best sequence of labels during decoding, the Viterbi algorithm is employed to decode the label sequence efficiently by maximizing the score $s(\mathbf{x}, \tilde{\mathbf{y}})$:

$$\mathbf{y}^* = \arg \max_{\tilde{\mathbf{y}} \in \mathbf{Y}_{\mathbf{x}}} s(\mathbf{x}, \tilde{\mathbf{y}}). \quad (13)$$

4 Experiments

4.1 Datasets

To analyze the effectiveness of our model for keyphrase extraction, we conduct comparative experiments on two scientific publication datasets provided by Caragea *et al.* [3], which are from two top-tier machine learning conferences: ACM Knowledge Discovery and Data Mining (KDD) and ACM World Wide

Web (WWW). Each dataset consists of the research paper titles, abstracts and corresponding author manually labeled keyphrases (gold standard).

A detail description of datasets is summarized in Table 1, containing the total number of abstracts and keyphrases in the original dataset ($\#Abs/\#KPs(All)$), the number of abstracts for which at least one author-labeled keyphrase could be located and the total number of keyphrases located ($\#Abs/\#KPs(Locatable)$), the percentage of keyphrases not present in the abstracts (MissingKPs), the average number of keyphrase per paper (AvgKPs), and the number of keyphrases with one, two, three and more than three tokens found in these abstracts.

4.2 Evaluation Metrics

Almost all previous works on keyphrase extraction use precision (P), recall (R) and F1-score ($F1$) to evaluate the results. Hence, we also keep our evaluation metric consistent. P , R and $F1$ are defined as follows:

$$P = \frac{\#_c}{\#_e}, R = \frac{\#_c}{\#_s}, F1 = \frac{2PR}{P + R} \quad (14)$$

where $\#_c$ is the number of correctly extracted keyphrases predicted by model, $\#_e$ is the total number of keyphrases predicted by model and $\#_s$ is the total number of standard keyphrases labeled by author.

In our experiments, we partition the dataset into three groups using tenfold cross-validation: Onefold is used as the testing data; Onefold is used as the validating data; the remaining eightfolds are used as the training data. We report the average results of tenfold cross-validation.

Table 1. Statistics of the two benchmark datasets.

Dataset	$\#Abs/\#Kps(All)$	$\#Abs/\#Kps(Loc.)$	MissingKPs (%)	$\#AvgKPs$	$\#unigrams$	$\#bigrams$	$\#trigrams$	$\#> trigrams$
KDD	365/1471	315/719	51.12	4.03	363	853	189	66
WWW	425/2073	388/904	56.39	4.87	680	1036	247	110

4.3 Implementation Details

In the experiment, we divide keyphrases into two categories: simple keyphrase (1-gram, referred to as SK) and complicated keyphrase (n -grams ($n > 1$), that is composed by two or more than two words, referred to as CK). Each word in dataset is labeled with non-keyphrase (O), simple keyphrase (SK) or complicated keyphrase (CK). For the complicated keyphrase, B-CK, M-CK and E-CK correspond to the beginning, middle and end word of CK, respectively. Thus, the number of labels is set to $k = 5$.

In order to eliminate the negative effects of different text formats, we convert the input text to lowercase in data pre-processing and employ two binary lexical features: whether the word contains digits or punctuation, which is similar to the

works [11, 15], We use the 50-dimensional embeddings pre-trained by GloVe¹ [18] on two corpora: Wikipedia2014² and Gigaword5³. In addition, some parameters of model are empirically set as follows: the hyper-parameter in Eq. (8) $\lambda = 0.6$, and the trainable parameters in Eq. (3) $\vec{W}_m$ and $\overleftarrow{W}_m \in \mathbb{R}^{50 \times 50}$, $b_m \in \mathbb{R}^{50}$. Dropout is applied to all GRU recurrent units on the input and output connections to avoid over-fitting, with a keep rate of 0.6 in the training procedure.

4.4 Comparative Methods

To evaluate the performance of our method, we compare our method with four state-of-the-art keyphrase extraction methods, as follows:

- **KEA** [24], which employs a supervised Naïve Bayes classifier to extract keyphrases using only two features: TF-IDF (*i.e.*, term frequency-inverse document frequency) of a phrase and the distance of a phrase from the beginning of a document (*i.e.*, its relative position).
- **CeKE** [3], which also uses a Naïve Bayes classifier for extracting keyphrases from research papers embedded in citation networks. This work designs some novel features for keyphrase extraction based on citation context information and uses them in conjunction with traditional features which have been widely used in the previous supervised works of keyphrase extraction.
- **EK-CRF** [9], which is the state-of-the-art traditional supervised method, and uses the CRF algorithm based on sequence labeling to extract keyphrases from research papers. This method incorporates the expert-knowledge and domain-specific hints.
- **CopyRNN** [16], which is the first to employ sequence-to-sequence (Seq2Seq) framework with attention and copy mechanisms to generate keyphrases. This method is able to predict absent keyphrases that do not appear in the target document.
- **M-CRF^s**, which is the simplified model of our complete model MLM-CRF. M-CRF^s only uses the attention at sentence level. That is, the coverage of attention mechanism in memory network only depends on the length of the sentence, which the current word belongs to. In this model, the output memory vector and attention weight are calculated by the Eq. (5) and (6), respectively.
- **M-CRF^d**, which is the another simplified model of our complete model MLM-CRF. M-CRF^d only calculates attention weight between the current word and the whole input document. In this model, the output memory vector and attention weight are computed by the Eq. (5) and (7), respectively.

4.5 Comparison with Supervised Prediction Methods

Table 2 shows the comparison of results of our model with other state-of-the-art supervised approaches. From Table 2, we can see that the overall results of KDD

¹ <http://nlp.stanford.edu/projects/glove/>.

² <https://code.google.com/archive/p/word2vec/>.

³ <https://catalog.ldc.upenn.edu/LDC2011T07>.

dataset is better than those of WWW, which is consistent with percentage of keyphrases not present in the abstracts (MissingKPs) in given research papers (MissingKPs = 51.12% on KDD, 56.39% on WWW), as given in Table 1. The benefit is that the experiment can reflect real application environment.

We first conduct experiments to compare the MLM-CRF with its two simplified models M-CRF^s and M-CRF^d. As the results given in Table 2, MLM-CRF gets the best results in terms of performance measures, and M-CRF^d achieves better results than M-CRF^s. These results indicate that long-range and more contextual information is more conducive to keyphrase extraction. More specifically, the contextual information captured by M-CRF^d is longer than by M-CRF^s, and MLM-CRF can capture the contextual information in the sentence level and document level.

Secondly, we discuss the comparison of our model MLM-CRF with other comparative keyphrase prediction methods, including KEA, CeKE, EK-CRF and CopyRNN. As given in Table 2, MLM-CRF outperforms all comparative methods on two datasets, and even the M-CRF^d has a margin over the best performing extraction method CRF on the two test datasets. It is also worth mentioning that CeKE includes features based on the document-citation network, EK-CRF designs complex features integrating expert-knowledge and domain-specific hints during keyphrase extraction, and CopyRNN can generate absent keyphrases, whereas our model does not need to use extra knowledge, design complex features, and can not predict absent keyphrases.

In conclusion, the MLM-CRF can capture automatically much useful information from the source text for keyphrase extraction. Thus, our model gets the best results in terms of the performance measures, indicating that our method indeed outperforms the other approaches on all two datasets.

Table 2. Comparison of the proposed models with other approaches

Method	KDD			WWW		
	Precision	Recall	F1-score	Precision	Recall	F1-score
KEA [24]	0.1551	0.3278	0.2105	0.1549	0.3182	0.2084
CeKE [3]	0.2174	0.3905	0.2793	0.2251	0.2519	0.2377
EK-CRF [9]	0.4068	0.2162	0.2823	0.3689	0.194	0.2547
CopyRNN@5 [16]	0.2221	0.4926	0.3062	0.1907	0.3993	0.2581
M-CRF ^s	0.3438	0.2354	0.2794	0.3276	0.2059	0.2528
M-CRF ^d	0.3597	0.2542	0.2979	0.3551	0.2129	0.2662
MLM-CRF	0.3787	0.2771	0.32	0.3251	0.2417	0.2773

4.6 Comparison in Different Types of Keyphrases

We compare our different models in two different types of keyphrases: SK (including only single word, *i.e.*, 1-gram) and CK (including several consecutive words,

i.e., n -gram, $n \geq 2$). We first compare the MLM-CRF with its two simplified models M-CRF^s and M-CRF^d in SK and CK on both KDD and WWW datasets, respectively. As the results given in Table 3, both in SK and in CK, MLM-CRF gets the best results in terms of the performance measures, and M-CRF^d achieves better results than M-CRF^s on two datasets. These results indicate that long-range and more contextual information is more conducive to keyphrase extraction.

Table 3. Comparison of our different models in different type keyphrases.

Methods	KDD						WWW					
	SK			CK			SK			CK		
	<i>P</i>	<i>R</i>	<i>F1</i>	<i>P</i>	<i>R</i>	<i>F1</i>	<i>P</i>	<i>R</i>	<i>F1</i>	<i>P</i>	<i>R</i>	<i>F1</i>
M-CRF ^s	0.3679	0.1235	0.1849	0.3382	0.3059	0.3212	0.3534	0.1868	0.2444	0.3092	0.2247	0.2603
M-CRF ^d	0.3180	0.1444	0.1986	0.3734	0.3235	0.3466	0.3659	0.2009	0.2594	0.3460	0.2247	0.2725
MLM-CRF	0.4589	0.1843	0.2630	0.3720	0.3520	0.3617	0.3453	0.2283	0.2749	0.3090	0.2548	0.2793

Secondly, we compare the growth performance from the simplified method M-CRF^s to MLM-CRF in SK and CK. From Table 3, we can see that the growth of F1-score is 0.0781 for SK and 0.0450 for CK on KDD, and is 0.0305 for SK and 0.0190 for CK on WWW. It is obvious enough that the growth of F1-score in SK is more than that in CK on both KDD and WWW datasets. We can obtain the similar growth trends of Precision and Recall in SK and CK on two datasets. These results show that the 1-gram keyphrases have a stronger long-distance dependencies in text sequence than the n -gram ($n \geq 2$) keyphrases, which are more dependent on the local structural information. The experimental setup might be able to explain the main reason for these results. More specifically, for identifying keyphrases labeled by CK using the CRF model, different labels of CK are restricted, while for identifying keyphrases labeled by SK, the single SK label is totally unrestricted. For example, the B-CK label must be followed by the M-CK or E-CK label in the experiments.

Finally, we discuss the performance of our models in different types of keyphrases SK and CK. As the results given in Table 3, we can see that our models can obtain better performance in CK than that in SK. The main reason may be that the percentage of 1-gram keyphrases in all keyphrases is significantly less than the percentage of n -gram ($n \geq 2$) keyphrases.

5 Conclusions

In this paper, we proposed a multi-level memory network with CRFs named MLM-CRF for extracting keyphrases from scientific research papers. In particular, we first extended the input memory of the memory network with two different levels (*i.e.*, sentence level and document level) to capture the long-range contextual information hidden in text data. We then employed the CRF model to

capture the structural dependencies between adjacent words in text sequence and determine whether a candidate phrase is a keyphrase. Our experimental results have shown that the proposed model MLM-CRF can significantly outperform the state-of-the-art supervised prediction approaches (including three extraction methods and one generation method) on both WWW and KDD datasets. In future, we plan to explore the more effective attention mechanism for taking much less computing costs in encoding the long document.

Acknowledgements. This work was partially supported by grants from the National Natural Science Foundation of China (Nos. U1933114, 61573231) and Open Project Foundation of Intelligent Information Processing Key Laboratory of Shanxi Province (No. CICIP2018004).

References

1. Alzaidy, R., Caragea, C., Giles, C.L.: Bi-LSTM-CRF sequence labeling for keyphrase extraction from scholarly documents. In: Proceedings of WWW, pp. 2551–2557 (2019)
2. Bhaskar, P., Nongmeikapam, K., Bandyopadhyay, S.: Keyphrase extraction in scientific articles: a supervised approach. In: Proceedings of COLING, pp. 17–24 (2012)
3. Caragea, C., Bulgarov, F., Godea, A., Das Gollapalli, S.: Citation-enhanced keyphrase extraction from research papers: a supervised approach. In: Proceedings of EMNLP, pp. 1435–1446 (2014)
4. Chen, W., Gao, Y., Zhang, J., King, I., Lyu, M.R.: Title-guided encoding for keyphrase generation. In: Proceedings of AAAI, pp. 6268–6275 (2019)
5. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: Proceedings of NIPS (2014)
6. Dai, Z., et al.: Transformer-XL: attentive language models beyond a fixed-length context. In: Proceedings of ACL, pp. 2978–2988 (2019)
7. Florescu, C., Caragea, C.: Positionrank: an unsupervised approach to keyphrase extraction from scholarly documents. In: Proceedings of ACL, pp. 1105–1115 (2017)
8. Gollapalli, S.D., Caragea, C.: Extracting keyphrases from research papers using citation networks. In: Proceedings of AAAI, pp. 1629–1635 (2014)
9. Gollapalli, S.D., Li, X.L., Yang, P.: Incorporating expert knowledge into keyphrase extraction. In: Proceedings of AAAI, pp. 3180–3187 (2017)
10. Hasan, K.S., Ng, V.: Automatic keyphrase extraction: a survey of the state of the art. In: Proceedings of ACL, pp. 1262–1273 (2014)
11. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint [arXiv:1508.01991](https://arxiv.org/abs/1508.01991) (2015)
12. Kumar, A., et al.: Ask me anything: dynamic memory networks for natural language processing. In: Proceedings of ICML, pp. 1378–1387 (2016)
13. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of ICML, pp. 282–289 (2001)
14. Linzen, T., Dupoux, E., Goldberg, Y.: Assessing the ability of LSTMs to learn syntax-sensitive dependencies. In: Transactions of ACL, pp. 521–535 (2016)

15. Liu, F., Baldwin, T., Cohn, T.: Capturing long-range contextual dependencies with memory-enhanced conditional random fields. In: *Proceedings of IJCNLP*, pp. 555–565 (2017)
16. Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., Chi, Y.: Deep keyphrase generation. In: *Proceedings of ACL*, pp. 582–592 (2017)
17. Miller, A., Fisch, A., Dodge, J., Karimi, A.H., Weston, J.: Key-value memory networks for directly reading documents. In: *Proceedings of EMNLP*, pp. 1400–1409 (2016)
18. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: *Proceedings of EMNLP*, pp. 1532–1543 (2014)
19. Sterckx, L., Caragea, C., Demeester, T., Devellder, C.: Supervised keyphrase extraction as positive unlabeled learning. In: *Proceedings of EMNLP*, pp. 1924–1929 (2016)
20. Sukhbaatar, S., Szlam, A., Weston, J., Fergus, R.: End-to-end memory networks. In: *Proceedings of NIPS*, pp. 2440–2448 (2015)
21. Tang, Y., et al.: Qalink: enriching text documents with relevant Q&A site contents. In: *Proceedings of CIKM*, pp. 1359–1368 (2017)
22. Wang, F., Wang, Z., Wang, S., Li, Z.: Exploiting description knowledge for keyphrase extraction. In: *Proceedings of PRICAI*, pp. 130–142 (2014)
23. Weston, J., Chopra, S., Bordes, A.: Memory networks. In: *Proceedings of ICLR* (2015)
24. Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: Kea: practical automated keyphrase extraction. In: *Proceedings of ACM DL*, pp. 254–255 (1999)
25. Yang, D., Wang, S., Li, Z.: Ensemble neural relation extraction with adaptive boosting. In: *Proceedings of IJCAI*, pp. 4532–4538 (2018)
26. Zhang, C., Wang, H., Liu, Y., Wu, D., Liao, Y., Wang, B.: Automatic keyword extraction from documents using conditional random fields. *J. Comput. Inf. Syst.* 4(3), 1169–1180 (2008)
27. Zhang, Y., Chang, Y., Liu, X., Gollapalli, S.D., Li, X.: Mike: keyphrase extraction by integrating multidimensional information. In: *Proceedings of CIKM*, pp. 1349–1358 (2017)
28. Zhao, J., Zhang, Y.: Incorporating linguistic constraints into keyphrase generation. In: *Proceedings of ACL*, pp. 5224–5233 (2019)