IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 5, 2018, accepted July 27, 2018, date of publication August 16, 2018, date of current version September 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2865589

Keyphrase Generation Based on Deep Seq2seq Model

YONG ZHANG AND WEIDONG XIAO

Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha 410073, China

Corresponding author: Yong Zhang (zhangyong_nudt@sina.com)

This work was supported by NSFC under Grant 71690233 and Grant 71331008.

ABSTRACT Keyphrase can provide highly summative information which can help us improve information utilization efficiency in the era of information overload. Though previous researches about keyphrase generation have provided some workable solutions, they generate keyphrase by ranking and selecting meaningful words from the source text. These approaches belong to an extractive method, by which they cannot effectively use semantic meaning of the source text, and are unable to generate keyphrases which do not appear in the source text. So we propose a sequence-to-sequence framework with attention mechanism, copy mechanism, and coverage mechanism, which can effectively deal with the above-mentioned drawbacks. The experimental results on five data sets reveal that our proposed model can achieve a better performance than the traditional extraction approaches and can also generate absent keyphrases which do not appear in the source text.

INDEX TERMS Abstraction, seq2seq, attention mechanism, copy mechanism, coverage mechanism.

I. INTRODUCTION

A keyphrase is a short, summative text which can express the main information of longer text. The keywords in academic articles are a typical case, which can provide core information about the related article. In our work, we don't make a clear distinction between ''keyword'' and ''keyphrase'', all of which mean the same thing. In addition, it is worth mentioning that ''keyphrase'' and ''keyword'' may contain multiple words. High quality keyphrases would play a key role in understanding, searching and organizing text content. Therefore, there have been many studies focused on ways of automatic keyphrase generation [1], [2]. Due to public accessibility, many academic article datasets have been used for evaluating performance of keyphrase generation. Similarly to them, we also focus on using academic article datasets for keyphrase generation.

At the moment, there are two main approaches about keyphrase generation: extraction and abstraction. Extractive methods get keyphrases taken directly from the source text, and this approach has been widely used, such as text categorization [3], text summarization [4] and so on. Most of keyphrase extraction algorithms consists of two steps. First, they get a list of candidates, then rank the list of candidates based on their importance to the source text. The ranking algorithms can be divided into supervised and unsupervised machine learning methods [5], [6].

However, keyphrase extraction approach has two major drawbacks. First, this approach can only generate keyphrases which have appeared in the source text, and it is not able to generate significative keyphrases with a slight difference relatively to the source text. In general, authors of academic articles commonly summarize source text and generate keyphrases based on their personal understanding, more than using a piece of source text directly. Meng *et al.* [2] use **absent keyphrase** and **present keyphrase** to distinguish whether the keyphrase appears in the source text. We use the same calls in the rest of this paper. Second, it is hard to rank keyphrase condidates. The previous approaches often adopt some statistical learning methods to capture important statistical features. However these statistical features can only discover important words on the statistics, and are unable to reveal the full semantics of the source text.

Another important approach of keyphrase generation is the abstractive method, which is similar to human assigning keyphrases. Given a source text, both would first read the text to understand the basic meaning of context, then digest its essential content to generate keyphrases. Due to the difficulty of abstractive methods, most of past works have been extractive. However, the recent success of sequence-to-sequence framework, also named as seq2seq, has made abstractive methods viable. The earliest seq2seq framework is used in machine translation [7], [8]. Subsequently, it is

quickly extended to other fields, such as text summarization [10], speech recognition [12] and video captioning [13]. The seq2seq framework includes two parts, encoder and decoder. The encoder is responsible for encoding the source text and obtaining its semantic representation, then decoder generates final results based on the semantic representation of the source text. Though there are some researches on keyphrase generation based on seq2seq framework [2], [14], they exhibit some undesirable behavior such as being unable to deal with out of vocabulary(OOV) words, and less considering correlation among generated keyphrases.

In this paper, we employ some operating mechanisms based on basic seq2seq framework to handle the above two issues. First, to effectively express the semantic features of the source text, we use the Gated Recurrent Unit(GRU) to compress the semantic information into a series of dense vectors. Second, we use attention mechanism [21] to strengthen the correspondence between keyphrases and different parts of the source text. Furthermore, in order to reduce the impact of OOV problem, we incorporate copy mechanism [11] which allows our model to generate out of vocabulary words. Finally, we use coverage mechanism [26] to establish correlation among generated keyphrases. Through the above architecture, our proposed model can not only generate keyphrases based on the semantic information of the source text, but also handle OOV problem and build the correlation between generated keyphrases.

II. RELATED WORK

A. EXTRACTIVE KEYPHRASE GENERATION

The keyphrase can use several words to describe a subject or a subtopic about documents. The extractive keyphrase generation is one of two keyphrase generation approaches. Because the extractive methods are relatively simple, and can get general results. Many extractive methods have been proposed. The typical extractive method can be broken into the following two steps: generating candidates and choosing.

The first step is to extract some keyphrase candidates and make up the list of keyphrase candidates. We would usually make the length of the list exceed the number that we want to extract a lot, because more quantity means more correct candidates. The method of extracting candidates mainly includes sequence match [6], [15], and extracting important n-grams phrases [16].

The second step is ranking the candidates based on the relationship between candidates and the source text, and the top candidates would be chosen as final results. The researches about ranking algorithms are the focus in the entire extractive keyphrase generation algorithms. The machine learning methods are widely employed, and they can be divided into two types: supervised and unsupervised. The supervised methods treat this issue as a binary classification task, and sort the candidates' list based on the positive scores [17], [18], [34]. The unsupervised methods mainly include exploring central nodes of text graph [19], topical clusters [5] and so on.

In addition to the general two-phase step, some other studies treat the extractive keyphrase generation as an entirety. Liu *et al.* [20] uses a alignment model to learn conversion mode from the source text to target keyphrases. Their model can handle with the problem of inconsistency vocabulary about the source text and target keyphrases. Zhang *et al.* [14] uses recurrent neural network to build sequence labeling model, which has been used to extract keyphrases from tweets. Although extractive approach is very simple, this approach can't fully understand semantic information of the source text and be unable to generate the keyphrases which don't appear in the source text.

B. ABSTRACTIVE KEYPHRASE GENERATION

Abstractive keyphrase generation is another important keyphrase generation approach. With the success of sequence-to-sequence framework, abstractive methods are gradually over extractive methods, and can generate the results more in the line of human expectations. The core of sequence-to-sequence (seq2seq) framework is the encoder-decoder architecture. Seq2seq framework uses encoder to encode the source text and gets the semantic presentation of the source text. Then, decoder is responsible for generating target texts based on presentation of the source text. In our work, seq2seq and encoder-decoder express the same meaning, and both can be referred as a sequence-to-sequence learning framework, which is first proposed by Sutskever *et al.* [7] to deal with the machine translation problems. It can fit many natural language processing tasks and get great success because of the power of dealing with variable-length text sequences [8], [10], [11].

There have been many researches about how to use different strategies to improve the performance of seq2seq models. Some researches use the attention mechanism, which is a soft alignment method and allow model to automatically map the output to different parts of the input. Bahdanau *et al.* [21] first uses attention mechanism in seq2seq framework, and significantly improves the model's performance in machine translation task. Luong *et al.* [22] studies the improved approach about attention mechanism from local and global perspectives respectively.

In order to fully reflect the source text information, some researchers directly copy the fragments from the source text as a part of the target text. This method is called copy mechanism. By integrate copy mechanism into the basic seq2seq framework, we can handle with some out of vocabulary (OOV) problems and make the predictions more accurate. It is worth mentioning that the copy mechanism looks a bit like the extractive approach. So the fused model is a hybrid between extraction and abstraction. Vinyals *et al.* [23] uses the Pointer-Network to achieve the reproduction of OOV words, and their work has been first applied to the machine translation task. Gu *et al.* [24] uses the Copynet to generate unknown words in text summarization task.

In addition, another important strategy is coverage mechanism, which can allow model to discourage repetition.

The abstractive models like to repeat themselves in many applications, such as text summarization and keyphrases generation. See *et al.* [11] uses coverage mechanism to handle with repetition problems in multi-sentence text sunmarization tasks. Similar to text summarization tasks, the goal of keyphrase generation is to generate highly-summative information. So the coverage mechanism can help model generate more concise results, and redundancy among generated keyphrases is less. However, few models about abstractive keyphrase generation consider this point.

Although seq2seq framework has been developed and has been widely used in many tasks, there are only few researches about abstractive keyphrase generation. Meng *et al.* [2] first proposed applying the seq2seq framework to keyphrase generation task. Their idea is similar to our work. They incorporate copy mechanism into the entire models, and this treatment enables the model to generate the OOV words. However, they don't consider the correlation among target keyphrases, so that the generated results are highly repeatable.

III. METHODOLOGY

In this section, we would introduce our proposed model. First, we would define the task of keyphrase generation. Then, we describe the calculation framework of sequence-to-sequence(seq2seq) and related calculation details. Finally, **attention mechanism**, **copy mechanism**, and **coverage mechanism** would been introduced respectively.

A. PROBLEM DEFINITION

The purpose of the keyphrase generation task is that the machine automatically generates several keyphrases to summarize the source text. We assume that dataset contains N samples, and the i-th sample $(x^{(i)}, k^{(i)})$ consists of one source text $x^{(i)}$ and m_i target keyphrases which can be described as $k^{(i)} = (k^{(i,1)}, k^{(i,2)}, \ldots, k^{(i,m_i)})$. Both source text $x^{(i)}$ and keyphrase $k^{(i,j)}$ are treated as word sequences as follows:

$$x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_{L_{x^{(i)}}}^{(i)}]$$
$$k^{(i,j)} = [y_1^{(i,j)}, y_2^{(i,j)}, \ldots, y_{L_{k^{(i,j)}}}^{(i,j)}]$$

$L_{x^{(i)}}$ and $L_{k^{(i,j)}}$ denote the length of word sequence $x^{(i)}$ and $k^{(i,j)}$ respectively. It is worth mentioning that the sequences of keyphrase are relatively short, and only contain a few words.

As described above, each sample consists of one source text $x^{(i)}$ and several keyphrase sequences $k^{(i)}$. We use seq2seq framework to explore the hidden patterns between the source text and target keyphrases. Each sample would be converted into several text-keyphrase pairs, and each pair only contains one source text and one keyphrase. We assume the i-th data sample contains m_i target keyphrases, and the converted data sample looks like: $[(x^{(i)}, k^{(i,1)}), (x^{(i)}, k^{(i,2)}), \ldots, (x^{(i)}, k^{(i,m_i)})]$. These pairs are linked by coverage mechanism. Our split method is a bit different from Meng *et al.* [2], which splits each data sample into m_i sub-samples, and they don't consider the relationship among these sub-samples.

Then we use seq2seq framework to learn the mapping from source text to target keyphrase. We use (x, y) and $[(x, y_1), (x, y_2), \ldots, (x, y_{m_i})]$ to denote a text-keyphrase pair and a set of related pairs respectively. Specifically, x is word sequence of the source text, and y_i is word sequence of i-th keyphrase.

B. SEQUENCE-TO-SEQUENCE FRAMEWORK

The basic framework of our proposed keyphrase generation model is an encoder-decoder architecture, which is shown in Figure 1. Since both encoder and decoder are sequence structures, our framework can be also called sequence-to-sequence(seq2seq). The core idea of this framework is using encoder to get semantic representation of the source text, a series of hidden representations $h = [h_1, h_2, \ldots, h_L]$. Then decoder generates keyphrases based on these hidden representations.

It is worth mentioning that the recurrent neural network (RNN) has a good effect on language modeling, and GRU and LSTM are the two most common variants in RNNs. Considering simplicity, we use bidirectional GRU and unidirectional GRU as our encoder and decoder respectively. The GRU is a variation of the recurrent neural network(RNN) with two gates, and u and r is called the update gate and reset gate

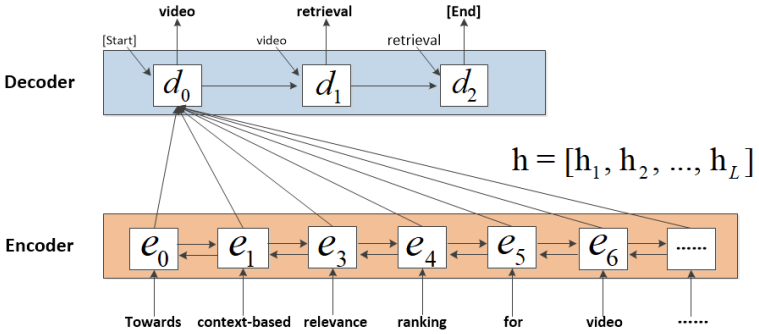

FIGURE 1. The basic encoder-decoder framework. Encoder and decoder are both sequence structures, and usually use variant recurrent neural network(RNN), such as LSTM or GRU. e_i and d_i present word vectors, and $h = [h_1, h_2, \ldots, h_L]$ is semantic representation of the source text.

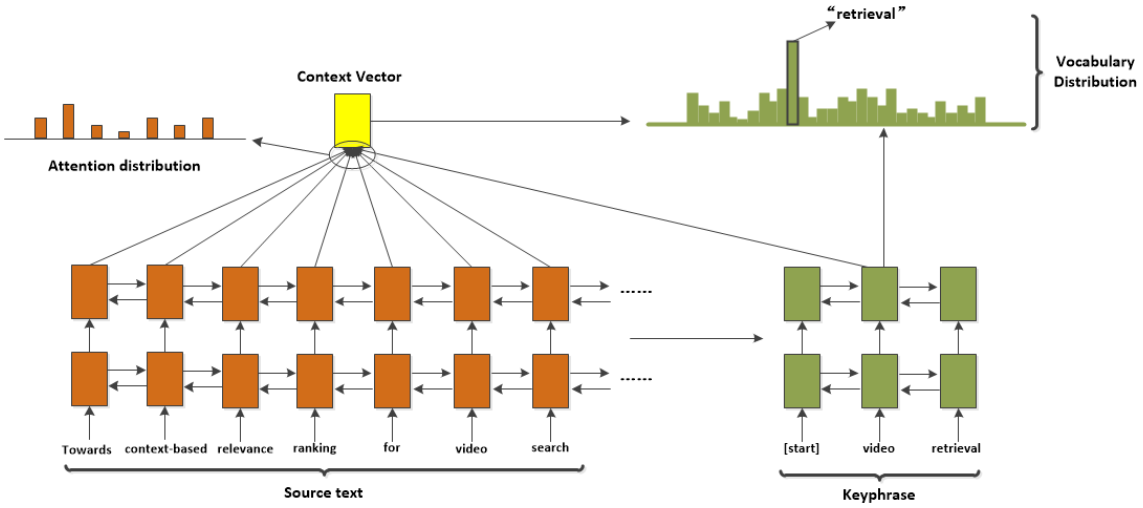

FIGURE 2. The seq2seq model with attention mechanism. The model uses two layers variant RNN as the encoder and decoder. The context vector is determined by semantic representation of the source text h and the recurrent state of decoder s_t. Then model uses context vector and s_t to compute vocabulary distribution P_{vocab}. It should be noted that this model can only generate words in the vocabulary, such as "retrieval".

respectively. The GRU is calculated as follows:

$$u_j = \sigma(W_{ux}x_j + W_{uh}h_{j-1} + b_u) \quad (1)$$
$$r_j = \sigma(W_{rx}x_j + W_{rh}h_{j-1} + b_r) \quad (2)$$
$$h_j^{'} = tanh(W_{hx}x_j + W_{hh}(r_j \bigodot h_{j-1} + b_h)) \quad (3)$$
$$h_j = (1 - u_j) \bigodot h_j^{'} + u_j \bigodot h_{j-1} \quad (4)$$

where W and b are GRU's learnable parameters, and h_j is the hidden representation vector at j-th timestep, and x_j is the input vector at corresponding timestep, and $\bigodot$ represents the Hadamard product. We can get the initialization vector of input x in a random way or Word2vec [25]. We prefer to utilizing Word2vec to initialize the input vector, and update it in the training process. This approach can accelerate model convergence.

In the framework, the bi-directional GRU is used to represent the source text, and it can be view as a combination of two uni-directional GRUs: forward and backward. The forward GRU computes sequence's current state based on current word embedding and previous sequence state. Another backward GRU computes sequence state in the reverse order from the last word to the first. The concatenation of two hidden representation is taken as presentation of the source text x:

$$h_j^f = f(x_j, h_{j-1}^f) \quad (5)$$
$$h_j^b = f(x_j, h_{j-1}^b) \quad (6)$$
$$h_j = [h_j^f, h_j^b] \quad (7)$$

where h_j^f and h_j^b indicate forward and backward sequence states at j-th timestep respectively, and h_j is the representation of the source text at j-th timestep, f is the GRU, '[]' represents vector concatenation.

C. ATTENTION MECHANISM

In keyphrase generation tasks, one source text produces multiple keyphrases, and each keyphrase focuses on different parts of the source text. So when the decoder uses semantic representation of the source text for decoding different keyphrases, the weight assigned to the source text should be different. To deal with the problem, our model incorporates attention mechanism in Bahdanau *et al.* [21], which is first introduced into language processing tasks and achieves good results in machine translation. The model with attention mechanism is shown in Figure 2.

In this work, we call the weight assigned to the source text as attention distribution. The attention distribution a^t is calculated as follows:

$$e_j^t = v^T tanh(W_h h_i + W_s s_t + b_{attn}) \quad (8)$$
$$a^t = softmax(e^t) \quad (9)$$

where v, W_h, W_S and b_{attn} are learnable parameters. h_i and s_t are hidden state of encoder and decoder respectively. a^t is attention distribution which can be viewed as a probability distribution over the source text, and it is able to tell decoder which parts should be more concerned in producing next word. Next, we utilize attention distribution a_t to calculate *context vector* h_t^*:

$$h_t^* = \sum_i a_i^t h_i \quad (10)$$

The context vector h_t^* can be considered as a fixed dimension representation of the source text when decoder generates the t-th word. Then the concatenation of context vector h_t^* and s_t is fed through two fully connection layer to calculate the vocabulary probability P_{vocab}:

$$P_{vocab} = softmax(V_2 tanh(V_1[s_t, h_t^*] + b_1) + b_2) \quad (11)$$

Where V_1, V_2, b_1 and b_2 are learnable parameters. Specially, V_2 is a matrix responding for changing the dimension to vocabulary size. P_{vocab} can be viewed a probability distribution over all words in the vocabulary, and it can provide us with the final distribution to predict word w:

$$P(w) = P_{vocab} \tag{12}$$

We use the negative log likelihood of the target word w_t^* as the loss at timestep t. So the $loss_t$ and $loss$ about whole sequence are:

$$loss_t = -logP(w_t^*) \tag{13}$$

$$loss = \frac{1}{T}\sum_{t=0}^{T} loss_t \tag{14}$$

D. COPY MECHANISM

We add **copy mechanism** to our basic seq2seq model with attention mechanism. We call the improved model as CopyRNN, which allows both copying words directly from the source text and generating words from a fixed vocabulary. CopyRNN model includes two modes: copy mode and generation mode. Our copy mode is very different from Copynet [24] because of its computational complexity, and is similar to Pointer-network [23]. CopyRNN is as shown in Figure 3.

We split all words into two parts: fixed vocabulary and OOV vocabulary. The fixed vocabulary only contains the most frequently occurring words, and it doesn't change with the different data samples. OOV vocabulary is different for different data samples. We use the extended vocabulary denote the union of fixed vocabulary and all words appearing in the source text. So the probability distribution over the extended vocabulary, also called final distribution, is calculated as follows:

$$P(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen})\sum_{i:w_i=w} a_i^t \tag{15}$$

Where p_{gen} can be viewed a soft switch to choose generating a word from the fixed vocabulary by probability distribution P_{vocab}, or copying a word directly from the corresponding source text based on attention distribution a^t. Note that if w is an out of vocabulary(OOV) word, that is w doesn't appear in fixed vocabulary, then P_{vocab} is zero. Similarly, if w doesn't appear in the source text, then $\sum_{i:w_i=w} a_i^t$ is zero. Futhermore, we use the context vector h_t^*, the decoder state s_t and the decoder input y_t to calculate the generation probability p_{gen}.

$$p_{gen} = \sigma(w_{ch^*}h_t^* + w_{cs}s_t + w_{cy}y_t + b_{gen}) \tag{16}$$

Where w_{ch^*}, w_{cs}, w_{cy} and b_{gen} are learnable parameters, σ is the sigmoid function which is used to normalize the probability distribution.

P_{vocab} and a^t are as described in (11) and (9) respectively. The calculation of loss function is in (12), (13), (14), but with a bit difference, and the probability distribution $P_{(w)}$ in (12) should be modified to (15).

E. COVERAGE MECHANISM

Repetition is a common problem in many seq2seq researches [11], [26], [27]. Especially, it is more important in Keyphrase

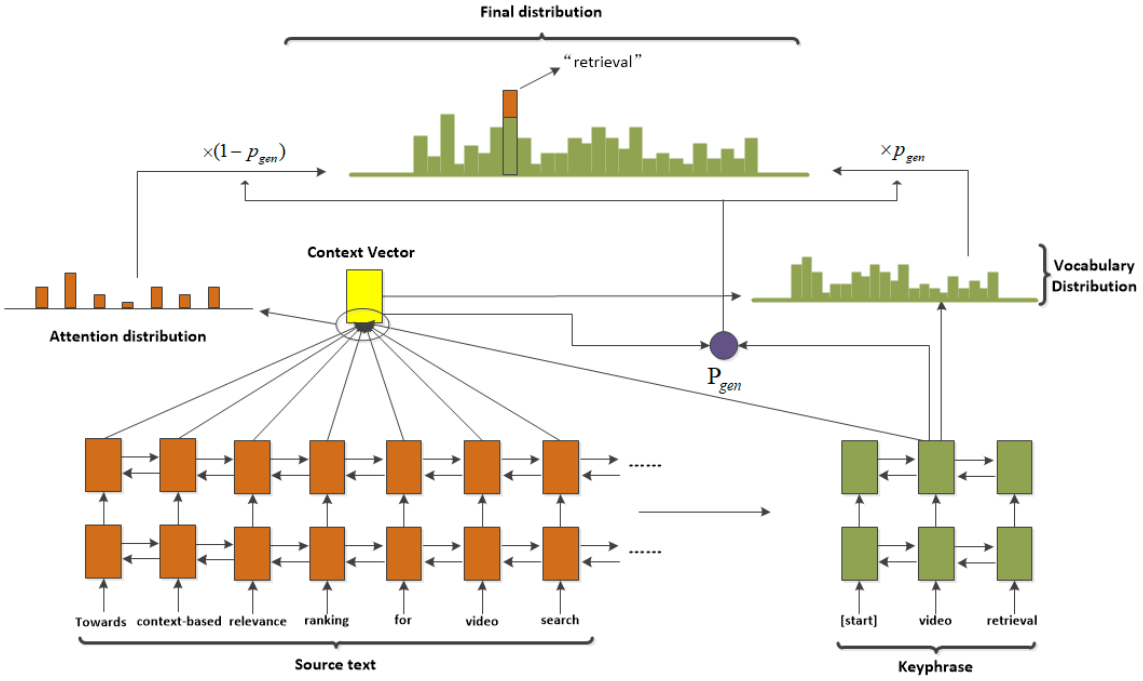

FIGURE 3. The seq2seq model with attention mechanism and copy mechanism. We call it CopyRNN. The final distribution consists of attention distribution and vocabulary distribution, and the weight relationship between them is determined by generation probability P_{gen}.

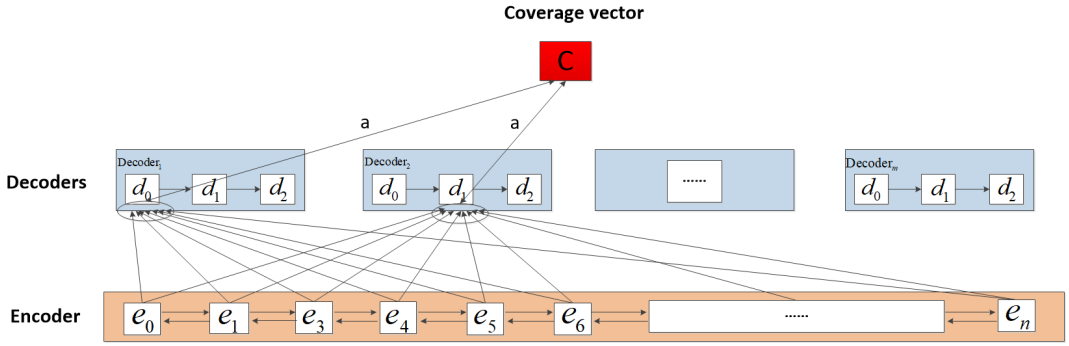

FIGURE 4. The seq2seq model with attention mechanism, copy mechanism and coverage mechanism. We call it CovRNN. The model contains an encoder and several decoders. Different decoders connect to each other via coverage vector, which can be seen as an accumulation of attention vector at different time steps.

generation tasks. Given a source text, our goal is to generate several keyphrases. In order to build relationship and reduce the information redundancy among these generated keyphrases, we add **coverage mechanism** to CopyRNN model and get a new improved seq2seq model CovRNN, which includes attention mechanism, copy mechanism and coverage mechanism. Our coverage mechanism is a bit like work of Tu *et al.* [26]. The details of CovRNN model is as shown in Figure 4.

In CovRNN model, we use a coverage vector c^t to memory the source text information which has already been concerned about. The coverage vector c^t is the cumulative sum of attention distributions over all previous decoder steps, and its calculation is as follows:

$$c^t = \sum_{t_s=0}^{t-1} a^{t_s} \tag{17}$$

Intuitively, c^t can be viewed as a memory vector, which is an unnormalized distribution upon words of the source text, and can describe the degree of coverage by accumulation of attention distributions over all previous decoder steps. It is mentioned that c^0 is zero vector, because none of the source text has been concerned at the first timestep.

The coverage mechanism is used to establish the relationship between different kehphrases. In general, the different keyphrases should concern about different parts of the source text, and the coincidence between them should be as low as possible. So we can regard coverage vector c^t as an extra input to help model improve the effectiveness of attention. The coverage mechanism can make later generated keyphrases pay more attention to the less concerned parts in the source text. We change (8) to:

$$e_j^t = v^T tanh(W_h h_i + W_s s_t + w_c c_i^t + b_{attn}) \tag{18}$$

where w_c is a learnable parameter vector, and its length is the same as v. In (18), we use model's previous decision(coverage vector c^t) to effect attention mechanism's current decision, that is telling model where to attend next. By coverage mechanism, we can make attention mechanism avoid repeatedly attending to the same parts of the source text.

Some researches [11], [26] show that coverage mechanism needs an extra loss function to penalize repeatedly attending to the same locations, otherwise it would be ineffective with no discernible reduction in repetition. We use the definition of [11]:

$$covloss_t = \sum_i min(a_i^t, c_i^t) \tag{19}$$

Our coverage loss function is different from Tu *et al.* [26], which deals with the repetition in MT(machine translation) tasks, however we focus on the keyphrases generation tasks. In MT, they assume the conversion ratio between source language text and target language text is a roughly one-one, and the final coverage vector would be penalized if it is away from 1. In keyphrase generation, we don't need uniform coverage, and we only penalize the overlap between each step's attention distribution and the current cumulative coverage vector for preventing repeated attention. Finally, we use hyperparameter λ to weight our coverage loss, and get a new composite loss function.

$$loss_t = -logP(w_t^*) + \lambda \sum_i min(a_i^t, c_i^t) \tag{20}$$

IV. EXPERIMENTS

In this section, we would introduce our experimental designs, including the description of training and testing datasets, implementation details, comparison models and evaluation metrics.

A. TRAINING DATASET

As far, there are several public datasets for keyphrase generation tasks. We use the dataset from Meng *et al.* [2], which is the largest one in all publicly available datasets. In the later description, we use KP to mark it. The KP dataset is collected by Meng *et al.*, and consists of a large number of high-quality scientific literatures about computer science.

In total, the KP dataset contains 567,830 articles. We use the same split method as Meng *et al.* [2] to split KP dataset. We randomly pick 40k articles, among which 20k articles are used as validation dataset to detect models' convergence in

training process. This way can help us to early stop training and improve efficiency. Another 20k articles are used to build a new testing dataset KP20k. Finally it is worth mentioning that we only train our models on the rest of KP dataset, 527,830 articles.

B. TESTING DATASETS

As shown above, we build a new dataset KP20k to test our proposed models. In addition, for evaluating our proposed model more comprehensively, we use other four scientific article datasets which are widely used in many natural language processing tasks. We use the union of title and abstract as our source text. The details of all testing datasets are as follows:

- **Inspec** [28]: This dataset includes 2,000 paper abstracts. We only use 500 papers in all papers as our testing dataset.
- **Krapivin** [29]: This dataset includes 2,304 papers with full text and keyphrases. We randomly choose 400 papers as our testing dataset.
- **NUS** [30]: We use all 211 papers as our testing dataset.
- **SemEval-2010** [31]: This dataset includes 288 papers from the ACM Digital Library. We use 100 articles among them as our testing dataset.
- **KP20k** [2]: This dataset contains titles, abstracts, and keyphrases about 20k articles in computer science.

It is worth noting that the amount of **Inspec, Krapivin, NUS, SemEval-2010** is too small to train a robust seq2seq model. So we use KP dataset to train our seq2seq model, and test on **Inspec, Krapivin, NUS, SemEval-2010, KP20** respectively.

C. IMPLEMENTATION DETAILS

These are 527,830 <text, keyphrases> paris for training, in which text represents concatenation of title and abstract, and keyphrases include several author-assigned keywords. Besides tokenization and lowercase, we don't use other ways to pre-processing data, and we believe that operating directly on the original data is favorable problem to solve.

We train three seq2seq models. The first model(RNN) is a basic seq2seq model with attention mechanism, and the second model(CopyRNN) includes attention mechanism and copy mechanism. The third model(CovRNN) contains attention mechanism, copy mechanism and coverage mechanism. For the three models, we set 50k most frequently appeared words as our vocabulary. The word embedding size is 256, and we use Skip-gram [25] to pre-train our word embedding model. The word embedding model is not fixed, and continues to update during seq2seq models' training process.

We use Xavier initializer to initialize our model parameters. The optimizer is set to Adam [32] with learning rate $= 1 * 10^{-4}$, gradient clipping $= 0.5$, and dropout rate $= 0.5$. The beam size $= 6$. The teacher forcing can cause the differences between training and testing, so we set teacher forcing rate $= 0.6$, and model generates keyphrases according to the test process by 0.4 rate. We use early-stopping strategy to control model's training process, that is the training would be stopped once convergence appears on the validation dataset.

Meng *et al.* [2] find that the seq2seq models tend to assign higher probabilities for shorter keyphrases, but many keyphrases consist of more than one word. To deal with this problem, they only keep the highest generation probability single-word phrase and remove the rest. In later experiments, we use the same treatment as them.

D. COMPARISON MODELS AND EVALUATION METRIC

We use two unsupervised algorithms (Tf-Idf, TextRank [35]) and one supervised algorithm Mani [34] as our baselines. In addition, we train three seq2seq models based on the use of attention mechanism, copy mechanism, coverage mechanism. The first seq2seq model(RNN) only contains attention mechanism, and the second model is a combination of the first model and copy mechanism. Furthermore, we add coverage mechanism to the second model and get the third model.

The first two models and the third model optimization goals(Loss function) are different, and latter's loss function has an extra item, see(13),(20). So to obtain our final model with coverage mechanism, we first train the first two models, and add the coverage mechanism to the second trained model. See *et al.* [11] tried to train the similar coverage model without covloss (19), hoping that attention mechanism can learn by itself not to attend to the same locations, but they found that this approach was not effective, and there was not discernible reduction in repetition.

We use F_1, precision and recall as our evaluation metrics to measure models' performance. The three evaluation metrics follow standard definition. The precision is defined as the number of correct predictions over the total number of predictions, and recall is defined as the number of correct predictions over the number of all samples. Furthermore, F_1 is used as a comprehensive indicator of models' effectiveness.

$$F_1 = \frac{2precision * recall}{precision + recall} \tag{21}$$

V. RESULTS AND DISCUSSION

In this section, we first compare our proposed models with other methods in terms of effectiveness. Then, we compare the difference between our models and extractive methods in predicting present keyphrase tasks. Third, we analysis the performance of copy mechanism and coverage mechanism in seq2seq framework. Finally, we discuss the balance of extraction and abstraction methods.

A. EXPERIMENTAL RESULTS

In this series of experiments, we use **Inspec, Krapivin, NUS, SemEval-2010, KP20k** as our benchmark datasets in the keyphrase generation tasks. It is worth noting that **Inspec, Krapivin, NUS, SemEval-2010** are too small to support the seq2seq models' train. So we use KP20k to train our seq2seq

TABLE 1. The performance of predicting keyphrases.

Method	Inspec		Krapivin		NUS		SemEval		KP20k	
	F_1@4	F_1@8	F_1@4	F_1@8	F_1@4	F_1@8	F_1@4	F_1@8	F_1@4	F_1@8
If-Idf	0.101	0.157	0.057	0.104	0.063	0.121	0.063	0.107	0.071	0.062
TextRank	0.091	0.146	0.042	0.091	0.061	0.120	0.059	0.101	0.061	0.064
Maui	0.032	0.035	0.163	0.151	0.164	0.171	0.033	0.032	0.171	0.165
RNN	0.081	0.058	0.132	0.084	0.151	0.107	0.141	0.102	0.171	0.163
CopyRNN	0.253	0.301	0.251	0.212	0.243	0.273	0.241	0.254	0.301	0.240
CovRNN	**0.264**	**0.312**	**0.261**	**0.242**	**0.253**	**0.284**	**0.251**	**0.262**	**0.311**	**0.252**

TABLE 2. The performance of predicting present keyphrases.

Method	Inspec		Krapivin		NUS		SemEval		KP20k	
	F_1@4	F_1@8	F_1@4	F_1@8	F_1@4	F_1@8	F_1@4	F_1@8	F_1@4	F_1@8
If-Idf	0.223	0.317	0.130	0.167	0.141	0.183	0.126	0.190	0.111	0.131
TextRank	0.224	0.271	0.171	0.152	0.183	0.191	0.171	0.181	0.174	0.142
Maui	0.037	0.041	0.247	0.216	0.242	0.271	0.041	0.037	0.271	0.234
RNN	0.084	0.061	0.133	0.087	0.154	0.152	0.143	0.112	0.179	0.191
CopyRNN	0.271	0.340	0.310	0.256	0.320	0.316	0.292	0.294	0.321	0.260
CovRNN	**0.280**	**0.350**	**0.312**	**0.257**	**0.321**	**0.340**	**0.301**	**0.295**	**0.323**	**0.270**

model (RNN, CopyRNN, CovRNN), then fine-tune them on the relevant dataset. In order to better analyse different models, our experiment would be divided into three types of tasks, predicting keyphrases, predicting present keyphrases, predicting absent keyphrases. We split present keyphrase and absent keyphrase based on whether it appears in the source text.

Table 1 provides the performance of three baseline models (Tf-Idf, ExpandRank, Maui) and three abstractive seq2seq models(RNN, CopyRNN, CovRNN) for predicting keyphrases tasks. For each model, the table lists its F_1 about top 4 and top8 predictions on the five benchmark datasets. We use the bold to highlight the best scores.

In order to measure overall performance of models, we don't filter keyphrases for evaluation in the task. In other words, the keyphrases for evaluation contain both present keyphrases and absent keyphrases. The experimental results show that, on the whole, the baseline models (Tf-Idf, TextRank, Maui) perform much worse than seq2seq models (RNN, CopyRNN, CovRNN). The reason is easy to understand. The baseline models belong to extractive methods which can't generate absent keyphrases, and seq2seq models belong to abstractive methods which can generate absent keyphrases. So the overall performance of baseline models is relatively poor. For example, for KP20k dataset, the best score in baseline models is 0.171 and 0.165, but the best score in seq2seq models reaches 0.311 and 0.252.

As for seq2seq models, the general RNN model with the attention mechanism doesn't perform as well as we expect. For example, for KP20k dataset, the F_1@4 score of the general RNN is 0.171. Relatively, the F_1@4 scores of CopyRNN and CovRNN are 0.301 and 0.311. We think that the main reason for this phenomenon lies in copy mechanism. The seq2seq model only concerns about the hidden semantics relation behind the source text and target keyphrases, so it tends to generate keyphrases which are too general to necessarily refer to the source text. The experimental results show that addition of coverage mechanism can also improve seq2seq model's performance, 0.301 vs 0.311 in F_1@4 metric for KP20k dataset.

B. PREDICTING PRESENT KEYPHRASES

The extractive methods are unable to generate absent keyphrases, so we can not compare extractive methods and abstractive methods fairly in general predicting keyphrases tasks. We utilize predicting present keyphrase tasks to compare performance of extractive methods and abstractive methods. In other words, we only consider present keyphrases for evaluation in keyphrase generation tasks. Similar to Table 1, Table 2 provides the performance of three baseline models(Tf-Idf, ExpandRank, Maui) and our proposed three seq2seq models(RNN, CopyRNN, CovRNN) in predicting present keyphrases tasks. For each method, we use F_1 at top 4 and top 8 to measure the performance. The best scores on five benchmark datasets are highlighted in bold.

In order to compare baseline models(Tf-Idf, TextRank, Maui) and seq2seq models(RNN, CopyRNN, CovRNN) fairly, we conduct experiments on predicting present keyphrase tasks. The results from Table 2 show that baseline models have more robust results on predicting present keyphrase tasks than on predicting keyphrase tasks. Overall, Maui performs best in all baseline models, and its F_1 at top 4 and top 8 reach 0.271 and 0.234 in KP20k dataset respectively. Although baseline models achieve good results, compared with seq2seq models, the latter is better, especially CopyRNN and CovRNN.

As for seq2seq models, similar to previous section, the RNN model with attention mechanism still does not achieve the expected results in predicting present keyphrase tasks. The models with copy mechanism(CopyRNN, CovRNN) are much better than RNN model, and comparison results about F_1 at top 4 and top 8 are 0.179 vs 0.321 and

Title: A **genetic algorithm** for the automated generation of small organic molecules: **Drug design** using an evolutionary algorithm

Abstract: Rational drug design involves finding solutions to large combinatorial problems for which an exhaustive search is impractical. Genetic algorithms provide a novel tool for the investigation of such problems. These are a class of algorithms that mimic some of the major characteristics of Darwinian evolution. LEA has been designed in order to conceive novel small organic molecules which satisfy quantitative structure-activity relationship based rules (fitness). The fitness consists of a sum of constraints that are range properties. The algorithm takes an initial set of fragments and iteratively improves them by means of crossover and mutation operators that are related to those involved in Darwinian evolution. The basis of the algorithm, its implementation and parameterization, are described together with an application in de novo molecular design of new retinoids. The results may be promising for chemical synthesis and show that this tool may find extensive applications in de novo drug design projects.

keyphrase: automated structure generation; **drug design**; **genetic algorithm**; molecular modeling; qsar; smiles; variable mapping

Result:
RNN: 1. drug; 2. **genetic algorithm**; 3. automated generation; 4. resolution method; 5. quantitative structure; 6. Main feature algorithm;7. novel tool; 8. general research; 9. initial set; 10. based rule
CopyRNN: 1. drug; 2. **genetic algorithm**; 3. **drug design**; 4. exhaustive search; 5. **molecular modeling**; 6. feature algorithm; 7. darwinian evolution; 8. quantitative structure; 9. organic molecule; 10. chemical syntheses
CovRNN: 1. **drug design**; 2. structure generation; 3. **genetic algorithm**; 4. automated driver; 5. **molecular modeling**; 7. feature engineering; 8. darwinan evolution; 9. application scene; 10. evolutionary algorithm

FIGURE 5. An example of predicting keyphrases by RNN, CopyRNN and CovRNN. The present keyphrases and correct keyphrases are highlighted in bold. The result show top 10 predictions for above three seq2seq models.

0.323, 0.191 vs 0.260 and 0.270 respectively. It is seems that copy mechanism is more important than we think, whether it is on predicting general keyphrase tasks, or on predicting present keyphrase tasks, because it can help seq2seq models generate keyphrases more closely matching the source text. Futhermore, we obverse that many keyphrases in our KP20k dataset contain OOV words. The general RNN model is unable to handle this situation, since it can only generate keyphrases consisted of words in vocabulary.

In addition, in order to build the connection between generated keyphrases from the same source text, we add coverage mechanism to CopyRNN model and get CovRNN model. The experimental results from Table 2 show that coverage mechanism can effectively improve seq2seq models, and F_1 of CopyRNN and CovRNN at top 4 and top 8 on KP20k dataset are 0.321 vs 0.323, and 0.260 vs 0.270. The reason may be that coverage mechanism can help seq2seq models remember past information of interest, so when the model generates a new keyphrase, the concern of past information of interest would be reduced. In this way, the content redundancy between keyphrases generated by seq2seq models would be small.

Figure 5 shows an example of predicting keyphrases by RNN, CopyRNN and CovRNN, and this example is about ''drug design'' topic. We use bold fonts to identify correct prediction results. We can observe that the three models can generate keyphrases which involve the topic of ''drug design''. However, RNN predictions are too general to be detect as keyphrases. Compared with RNN, CopyRNN can not only predict many detailed keyphrases like ''drug design'', but also generate OOV words, such as ''darwinian''. One the other hand, the results generated by CovRNN are less reproducible, and we can clearly observe that prediction results of CovRNN are much better than RNN and CopyRNN.

C. PREDICTING ABSENT KEYPHRASES

We expand the general predicting keyphrase tasks to predicting present keyphrase tasks and predicting absent keyphrase tasks. In the previous subsection, we compare baseline models with seq2seq models in predicting present keyphrase tasks. It is worth noting that predicting absent keyphrase is a very challenging task, as far as we know, only seq2seq models can handle this task. So we only perform the RNN, CopyRNN and CovRNN experiments and discuss the related results in this task.

Here, we use recall of top 10 and top 50 results to evaluate the performance of different seq2seq models. The recall metric can help us know how many keyphrases to be accurately predicted. Similarly, we use absent keyphrases in testing datesets to evaluate. The absent keyphrase predicting performance of RNN, CopyRNN and CovRNN is shown in Table 3. We use bold to indicate the best predictions.

TABLE 3. The performance of predicting absent keyphrases.

Dataset	RNN		CopyRNN		CovRNN	
	F_1@10	F_1@50	F_1@10	F_1@50	F_1@10	F_1@50
Inspec	0.032	0.063	0.045	0.102	**0.048**	**0.113**
Krapivin	0.096	0.158	0.115	0.191	**0.131**	**0.202**
NUS	0.047	0.089	0.059	0.118	**0.064**	**0.121**
SemEval	0.041	0.058	0.045	0.069	**0.049**	**0.073**
KP20k	0.085	0.143	0.125	0.191	**0.129**	**0.213**

From Table 3, we can see that the recall of RNN, CopyRNN and CovRNN can reach around 6.0%(10.2%), 7.8%(13.4%) and 8.4%(14.4%) of correct keyphrases at top 10(50) predictions. This performance shows that all three seq2seq models can capture hidden semantics behind the source text. In addition, CopyRNN model and CovRNN model perform much better than RNN model, and it reaches 30%(31.4%) and 40%(41.2%) on the recall metric of top 10(50). This huge improvement shows that copy machanism will still play a very important role in absent keyphrase prediction tasks. Then, with the advantage of generated keyphrases connection, the CovRNN model also outperforms the CopyRNN model, and it reaches 7.7%(7.5%).

From Figure 5, we can observe that absent keyphrases ''molecular modeling'' is correctly predicted by the last two seq2seq models. Although ''molecular modeling'' does not

appear in the source text, the models still apperceive the source text and paraphrase it to the target keyphrases. Although the correct number of predictions about CopyRNN model and CovRNN model in the example is the same, we can observe that the correlation between generated keyphrases in CopyRNN model is stronger than CovRNN model's, such as ''drug'' and ''drug design''.

D. DISCUSSION

The core goal of our work is to predict absent keyphrases based on ''understanding'' of the source text. It is a very challenging task, as far as we know, no existing methods except seq2seq models can handle this task. Therefore, we use the general seq2seq model with attention mechanism as our seq2seq baseline model. Considering the OOV and multiple keyphrases, we add copy mechanism and coverage mechanism to the general seq2seq model, and get CopyRNN and CovRNN. The experimental results show that CovRNN model gets the best results in keyphrase prediction tasks.

In order to deal with OOV problem, we add copy mechanism to the general seq2seq model. Related experiments show that the seq2seq model with copy mechanism is much better than not, and copy mechanism can significantly enhance keyphrase prediction accuracy. Intuitively, copy mechanism is more like an extractive method which selectively copies words from the source text. So we can view the seq2seq model with copy mechanism as a hybrid method of extraction and abstraction. It is worth noting that this hybrid approach is more in line with the habit of human. Considering the process by which humans summarize the source text to generate keyphrases, sometimes the generated keyphrases don't appear in the source text at all, and sometimes we choose some words from the source text as the generated keyphrase. In any case, encouraging the model to generate more abstractively like human, while retaining accuracy of keyphrase generation, copy mechanism would play an important role.

The source text and keyphrases are one-to-many relationships, and these generated keyphrases interact with each other. When we humans summarize keyphrases from the source text, we would consider the relationship between different keyphrases. Our goal is to make the generated keyphrases as complete as possible to express the meaning of the source text. At the same time, the meaning of different kehphrases is different as much as possible, that is, there is a low content redundancy between different keyphrases. So we introduce coverage mechanism in our work to establish links between generated keyphrases. The experimental results represent that this approach can effectively reduce the redundancy between generated keyphrases. Although coverage mechanism can establish the connection between generated keyphrases and make model concern different parts of the source text when generating different keyphrases, how to make the generated keyphrases more completely express meaning of the source text still is an unsolved problem.

VI. CONCLUSION AND FUTURE WORK

In this work, we proposed a seq2seq model with attention mechanism, copy mechanism and coverage mechanism for predicting keyphrase from the source text. Our model generates keyphrases based on the semantic understanding of the source text. It is able to deal with OOV problem by incorporating copy mechanism, and can reduce information redundancy between different keyphrases by merging coverage mechanism. Experimental results demonstrate the effectiveness of our proposed model for keyphrase generation tasks, including general keyphrase generation, present keyphrase generation and absent keyphrase generation.

In the future, the following two research directions need us to continue to expand:

- In this work, we use copy mechanism to deal with the OOV problem. This approach can make model generate more accurate details, and it belongs to an extraction method. Experiments show that this method has achieved very good results. Similarly, humans summarize keyphrases from the source text by the combination of extraction and abstraction. In the future, we will explore new ways of extraction and abstraction cooperation by studying the pattern of human summarization habit.
- We use coverage mechanism to establish the relationship between generated keyphrases, and choose top N candidates as final results. The number of generated keyphrases from different source texts should be different. It would be interesting to explore how to determine the number of generated keyphrases.

REFERENCES

[1] Z. Liu, P. Li, Y. Zheng, and M. Sun, ''Clustering to find exemplar terms for keyphrase extraction,'' in *Proc. Conf. Empirical Methods Natural Lang. Process.*, vol. 1. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 257–266.
[2] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi. (Apr. 2017). ''Deep keyphrase generation.'' [Online]. Available: https://arxiv.org/abs/1704.06879
[3] A. Hulth and B. B. Megyesi, ''A study on automatically extracted keywords in text categorization,'' in *Proc. 21st Int. Conf. Comput. Linguistics, 44th Annu. Meeting Assoc. Comput. Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006, pp. 537–544.
[4] R. Nallapati, F. Zhai, and B. Zhou, ''SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents,'' in *AAAI*, 2017, pp. 3075–3081.
[5] Z. Liu, W. Huang, Y. Zheng, and M. Sun, ''Automatic keyphrase extraction via topic decomposition,'' in *Proc. Conf. Empirical Methods Natural Lang. Process.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 366–376.
[6] T. T. N. Le, M. Le Nguyen, and A. Shimazu, ''Unsupervised keyphrase extraction: Introducing new kinds of words to keyphrases,'' in *Proc. Australas. Joint Conf. Artif. Intell.* Cham, Switzerland: Springer, 2016, pp. 665–671.
[7] I. Sutskever, O. Vinyals, and Q. V. Le, ''Sequence to sequence learning with neural networks,'' in *Proc. Adv. Neural Inf. Process. Syst.* 2014, pp. 3104–3112.
[8] K. Cho *et al.* (Sep. 2014). ''Learning phrase representations using RNN encoder-decoder for statistical machine translation.'' [Online]. Available: https://arxiv.org/abs/1406.1078
[9] S. Chopra, M. Auli, A. M. Rush, ''Abstractive sentence summarization with attentive recurrent neural networks,'' in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 93–98.

[10] A. M. Rush, S. Chopra, and J. Weston. (Sep. 2015). "A neural attention model for abstractive sentence summarization." [Online]. Available: https://arxiv.org/abs/1509.00685
[11] A. See, P. J. Liu, and C. D. Manning. (Apr. 2017). "Get to the point: Summarization with pointer-generator networks." [Online]. Available: https://arxiv.org/abs/1704.04368
[12] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 4945–4949.
[13] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence—Video to text," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4534–4542.
[14] Q. Zhang, Y. Wang, Y. Gong, and X. Huang, "Keyphrase extraction using deep recurrent neural networks on Twitter," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 836–845.
[15] M. Wang, B. Zhao, and Y. Huang, "PTR: Phrase-based topical ranking for automatic keyphrase extraction in scientific publications," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2016, pp. 120–128.
[16] O. Medelyan, I. H. Witten, and D. Milne, "Topic indexing with Wikipedia," in *Proc. AAAI WikiAI Workshop*, vol. 1, 2008, pp. 19–24.
[17] P. Lopez and L. Romary, "HUMB: Automatic key term extraction from scientific articles in GROBID," in *Proc. 5th Int. Workshop Semantic Eval.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 248–251.
[18] S. D. Gollapalli and C. Caragea, "Extracting keyphrases from research papers using citation networks," in *Proc. AAAI*, 2014, pp. 1629–1635.
[19] M. Grineva, M. Grinev, and D. Lizorkin, "Extracting key terms from noisy and multitheme documents," in *Proc. ACM 18th Int. Conf. World Wide Web*, 2009, pp. 661–670.
[20] Z. Liu, X. Chen, Y. Zheng, and M. Sun, "Automatic keyphrase extraction by bridging vocabulary gap," in *Proc. 15th Conf. Comput. Natural Lang. Learn.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 135–144.
[21] D. Bahdanau, K. Cho, and Y. Bengio. (Sep. 2014). "Neural machine translation by jointly learning to align and translate." [Online]. Available: https://arxiv.org/abs/1409.0473
[22] M.-T. Luong, H. Pham, and C. D. Manning. (Sep. 2015). "Effective approaches to attention-based neural machine translation." [Online]. Available: https://arxiv.org/abs/1508.04025
[23] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2692–2700.
[24] J. Gu, Z. Lu, H. Li, and V. O. K. Li. (Jun. 2016). "Incorporating copying mechanism in sequence-to-sequence learning." [Online]. Available: https://arxiv.org/abs/1603.06393
[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[26] Z. Tu. (Aug. 2016). "Modeling coverage for neural machine translation." [Online]. Available: https://arxiv.org/abs/1601.04811
[27] J. Suzuki and M. Nagata. (2016). "Cutting-off redundant repeating generations for neural abstractive summarization." [Online]. Available: https://arxiv.org/abs/1701.00138
[28] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 216–223.
[29] M. Krapivin, A. Autaeu, and M. Marchese, "Large dataset for keyphrases extraction," Dept. Inf. Eng. Comput. Sci., Univ. Trento, Trentino, Italy, Tech. Rep. DISI-09-055, 2009.
[30] T. D. Nguyen and M.-Y. Kan, "Keyphrase extraction in scientific publications," in *Proc. Int. Conf. Asian Digit. Libraries*. Berlin, Germany: Springer, 2007, pp. 317–326.
[31] S. N. Kim, O. Medelyan, M. Y. Kan, and T. Baldwin, "SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles," in *Proc. 5th Int. Workshop Semantic Eval. Assoc. Comput. Linguistics*, 2010, pp. 21–26.
[32] D. P. Kingma and J. Ba. (Dec. 2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980
[33] X. Wan and J. Xiao, "Single document keyphrase extraction using neighborhood knowledge," in *Proc. AAAI*, vol. 8, 2008, pp. 855–860.
[34] O. Medelyan, E. Frank, and I. H. Witten, "Human-competitive tagging using automatic keyphrase extraction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, vol. 3. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 1318–1327.
[35] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2004, pp. 1–8.

YONG ZHANG received the M.S. degree from the National University of Defense Technology, Changsha, Hunan, China, where he is currently pursuing the Ph.D. degree with the Science and Technology on Information System Engineering Laboratory. His research interests include natural language processing, machine learning, and sequence-to-sequence learning.

WEIDONG XIAO was born in Harbin, China, in 1968. He received the M.S. and Ph.D. degrees in management science and engineering from the National University of Defense Technology. He is currently a Professor with the Science and Technology on Information System Engineering Laboratory, National University of Defense Technology. His research interests include intelligence analytics, natural language processing, and knowledge graph.