**ISBN** 978-981-14-1684-2
**Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering**
(WCSE 2019 SUMMER)
*Hong Kong, 15-17 June, 2019, pp. 721-727*
doi: 10.18178/wcse.2019.06.107

# Keyphrase Generation with a Seq2seq Model

Pengfei Zhang[+], Dan Li[+] , Yuheng Wang and Yang Fang

National University of Defense Technology

**Abstract.** Keyphrases provide the core information of the source text, thus useful in many applications. Previous researches focus on extracting the keywords from the original document but miss those absent keyphrases unseen from the source. So we propose a generative model based on seq2seq RNN, which can generate both present and absent keyphrases by capturing the semantic information of the source. We adopt the large vocabulary trick to construct the target words corpus so as to improve the efficiency. We also introduce the feature-rich encoders to leverage the linguistic and statistical information in the source. Additionally, we incorporate the switching generator-pointer mechanism to extract those out-of-vocabulary words from the original document. To evaluate our model, we conduct two tasks, i.e., predicting present keyphrases and generating absent keyphrases, on real-life datasets. The results prove the effectiveness of our model as it outperforms the state-of-the-are models consistently and significantly.

**Keywords:** Keyphrase Generation; Seq2seq Model; Recurrent Neural Network

## 1. Introduction

Keyphrase generation plays a vital role in natural language processing (NLP) and information retrieval (IR). Keyphrases are the highly condensed words representing the core information of a given source text which could be used for further analysis and applications, for example, opinion mining [1], text categorization [25] and so on. In this paper, 'keyphrase' and 'keywords' are two interchangeable terms. Because of the various applications of keyphrases mentioned above, lots of models have been proposed to extract or generate keyphrases from the source text [12]. Because of public accessibility, like these proposed methods, we also focus on scientific publication datasets to testify the effectiveness of models.

To extract keyphrases, most existing models rely on two steps [12, 20]. First is to obtain a list of keyphrases candidates via n-grams with part-of-speech patterns or noun phrases [5, 23]. Second is to rank the candidates using machine learning approaches with manually-defined features [16, 18]. However, such two-step methods have two major drawbacks, one is that these methods can only extract keyphrases using words from the source (denoted as present keyphrases) while a large proportion of keyphrases may not match any continuous subsequence of the original document(denoted as absent keyphrases). Table I presents the percentage of present and absent keyphrases from commonly-used datasets. The other is that those models above tend to ignore the semantics of the source by only using machine learning features like TF and IDF.

Table 1: Percentage of Present and Absent Keyphrases in Public Datasets

| **Datasets SemEval** | **Inspec** | **Krapivin** | **NUS** | **SemEval** |
|---|---|---|---|---|
| **# Keyphrase** | 19,275 | 2,461 | 2,834 | 12,296 |
| **% Present** | 55.69 | 44.74 | 67.75 | 42.01 |
| **% Absent** | 44.31 | 52.26 | 32.25 | 57.99 |

To address the issues above, we propose a seq2seq model based on RNNs [2] [3], which is able to represent the semantics of the original document via low-dimensional vectors. Additionally, since it could

[+] Corresponding Authors :
*E-mail address*: hncszpf@163.com, 13317310232@163.com.

understand the original content, it is able to generate present and absent keyphrases from the vocabulary instead of the source text. We also incorporate those statistical features like TF and IDF and linguistic features like named-entity tags and parts-of-speech, with the word embeddings of the source text.

Basically, our model is a generative method, and we establish a vocabulary composed of frequent words with the appearance frequency over a certain threshold. However, it cannot generate the rare words which exist in the source document but not included in the vocabulary. We call those words as out-of-vocabulary (OOV) words. To solve the problem, we introduce a switching generator-pointer which could point the OOV words to the location in the source.

Finally, we could use the above seq2seq framework to capture the semantic or statistic information of the given document via a highly-compressed vector, no matter whether the keyphrases are present or absent in the source.

This work has three major contributions depicted as follow:

- We propose a seq2seq model to generate keyphrases no matter being present or absent, which could fully leverage the semantic information of the source document.
- We incorporate our model with a switching generator-pointer so as to extract those OOV words.
- We evaluate our model on real-life datasets and it achieves state-of-the-art experimental performance in comparison with other advanced alternatives.

Conference host: China Association of Sports Mathematics and Engineering, etc

In the following, we introduce the related work in Section 2, and then present the detailed mechanisms of our method in Section 3. Next, we introduce our experiment preparations in Section 4, along with the experiment results and analysis in Section 5. Finally, we conclude our findings in Section 6.

## 2. Related Work

Keyphrases summarize the source text with accurate and essential information. Previous models are extractive methods in essence, and could be generally separated into two steps.

The first step is to extract keyphrases candidates. Since the candidates will be further analysed and selected, a considerable number of candidates will be extracted so as to cover as many correct keyphrases as possible. To obtain keyphrases, previous works mainly extract word sequences that match some part-of-speech patterns like nouns or adjectives [10, 11], or extract noun phrases or n-grams [5, 14].

The second step is to rank a candidate to decide whether it is suitable to summarize the original text. Those candidates ranked highly will be chosen as keyphrases. It can be realized in a supervised or unsupervised fashion. In the supervised way, ranking the candidates is actually a binary classification issue with each candidate labeled as 'keyphrase' or 'non-keyphrase'. One needs to tag the training datasets as well. Multiple models are proposed to identify the candidates as keyphrases in the source text [5, 13, 22]. KEA [22] proposes to feed TF and IDF features and a phrase's first appearance to Naive Bayes. KEA++ [19] is an extension of KEA, which incorporates the KEA with semantic information behind the phrases from a corpora with specific domain, resulting in a better performance. In the unsupervised way, TF and IDF features are widely used in [7, 24]. The proportion of a topic and knowledge of a specific domain is utilized to rank the candidate phrases. TextRank [16] extracts keyphrases through PageRank values which are calculated by ranking the graphs obtained from the source. [12] proposed a model based on clustering which could ensure that the semantics of the source could be fully represented by the chosen keyphrases.

Other than the above commonly used two-step models, recently several models have been proposed to extract or generate the keyphrases in totally different ways. For instance, [11] utilized a word alignment model to learn a translation from source texts to keyphrases. Such mechanism alleviate the vocabulary gap between source and target. Nevertheless, it is unable to handle the semantic information. [23] introduce a RNN with joint-layer to extract keyphrases from tweets, however, it focuses on sequence labeling, thus unable to handle absent keyphrases.

## 3. Proposed Model

## 3.1. Seq2seq RNN with large vocabulary trick

The encoder of our model is a bidirectional GRU, while the decoder is a uni-directional GRU with the same hidden states as that of the encoder. We also apply an attention mechanism over the hidden states and finally a softmax layer to generate phrases over the target vocabulary. However if without limitation, the softmax layer is actually a full-connected layer producing a distribution over all the words of the source. No doubt it will be rather inefficient. In order to improve the computation cost, we adopt a large vocabulary trick (LVT) [6] which creates target word vocabulary according to the words' frequency in the source. The words with appearance frequency higher than a threshold will be selected into the target vocabulary. Moreover, such trick can help improve the model efficiency as it enables the model to focus only on the important words in the source. It is intuitive that in a summary most words come from the source text, so the LVT is well suited to the abstract summarization task.

## 3.2. Adding additional features to input sequence

Traditional seq2seq models adopted the word embeddings generated by word2vec technique as the input representations. However, it is insufficient as it ignores the other meaningful information like named-entity tags, parts-of-speech tags containing linguistic features, and statistical information like IDF and TF. To address such issue, we add the above linguistic and statistical information to embeddings of the words. Specifically, for the linguistic tags, similar as word embeddings, we embed them as look-up based representation matrices. For the TF and IDF which are continuous, we use one-hot embeddings to transfer them into discrete values as fixed binary numbers. By doing so, such statistical information could be mapped into representations matrix just like the linguistic tag information. At last, we simply concatenate those embeddings to obtain the final feature-rich embeddings of the input sequence of the encoder. The Figure 1 illustrates the process of adding additional features to the input sequence. As for the final output sequence of the decoder, we still use the word embeddings to identify the words.

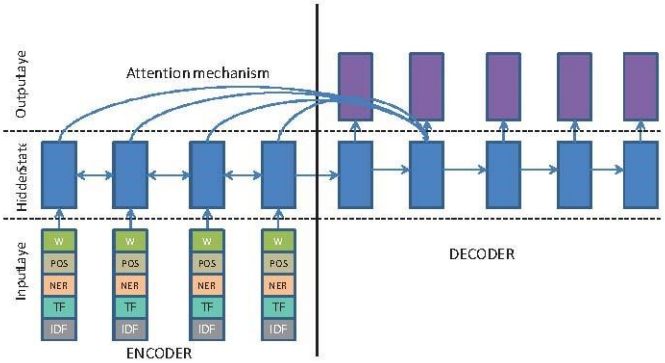


Fig. 1: Adding additional features to encoder

## 3.3. Switching generator-pointer

It is common that our target vocabulary created by LVT can not cover all the words in a summary, as it only selects words over a certain frequency threshold. Some rare words are likely to be missed bot important to some summaries. To solve the problem above, we introduce a switching generator-pointer to help the model point the location of the target words in the summary. The framework is illustrated in Fig 2. We add a switching mechanism to the encoder which will decide whether to generate a word from the vocabulary or to point to the source to extract the target word. In our mechanism, we set that generator is activated when the switch is on, and if the switch is off, the pointer will correspondingly work. We use a sigmoid function to model such gating mechanism at each time step and the mathematical formula is shown as follow:

$$P(f_i = 1) = \sigma(u^f \bullet (W_g^f g_i + W_e^f E_{i-1} + W_a^f a_i + b^f)),$$

in which $P(f_i = 1)$ is the probability of being switched on, $i$ is the time step, $\sigma$ is the activation function and we use sigmoid in our model. $g_i$ is the hidden states of the encoder and $E_{i-1}$ is the embeddings from the previous step. $a_i$ is the attention score and $b^f$ is the bias. $W_g^f$ , $W_e^f$ and $W_a^f$ are those switching parameters. To point to the location of the word in source, we use the attention distribution over the source text as follows:

$$P_i^c(j) \propto \exp(u^c \bullet (W_g^c g_{j-1} + W_e^c E_{i-1} + W_a^c g_j^d + b^c)),$$

where $P_i^c(j)$ is the probability of the decoder pointing to the location $j \in \{1, \ldots N\}$ of the source at time step $i$, and $N$ is the words of the source. $g_j^d$ is the hidden state of decoder on the location $j$. Hence we can obtain the pointer value of location $i$ in the summary that is sampled from $P_i^c(j)$:

$$l_i = \arg\max_j (P_i^c(j)), j \in \{1, \ldots N\}.$$

For the objective function, we optimize the following conditional log-likelihood function adding regularization:

$$\log P(\mathbf{t} \mid \mathbf{s}) = \sum_i h_i \log\{P(t_i \mid \mathbf{t}_{-i}, \mathbf{s}) P(f_i)\} + (1 - h_i) \log\{P(l_i \mid \mathbf{t}_{-i}, \mathbf{s})(1 - P(f_i))\},$$

in which $\mathbf{s}$ and $\mathbf{t}$ represents the source words and target words, respectively. $h_i$ is the indicator that if it is set to 0, the word at summary location $i$ is a OOV word. When testing, our model is able to automatically decide to use generator or pointer according to the computed switching probability $P(f_i)$. At each time step, we use the argmax of the subsequent probability of generating or pointing to produce the best output sequence.

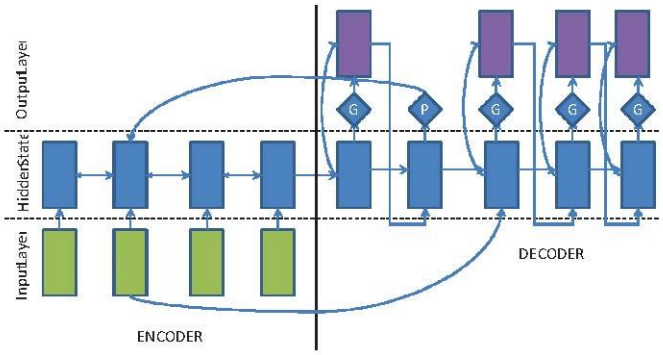


Fig. 2: Switching generator-pointer framework

## 4. Experiment Preparation

We first introduce the training and testing datasets, and then present the pre-process information and parameters in detail. Afterwards we introduce the evaluation metrics and the other state-of-the-are alternatives that we compare in the experiment.

### 4.1. Datasets

We focus on the web-available scientific publications as the training datasets since they contain the keywords or keyphrases assigned by the authors themselves. We directly adopt the datasets constructed by [15], which contains lots of scientific publications whose domain is computer science. Those publications are extracted from various reliable online libraries such as ACM library, Wiley, Web of Science and so on. Actually, the datasets above contain 567, 830 papers in total after removing the duplicates. To construct the validation datasets and testing datasets, we randomly choose 40,000 papers and evenly split it into validation datasets for checking the convergence and testing datasets denoted as KP20K. Consequently, our training datasets have 527, 830 papers in total.

To evaluate our model in a more comprehensive way, we also include other four scientific publication sources as testing datasets, namely, Inspec [5], NUS [17], Krapivin [9] and SemEval [8]. In specific, Inspec has 2, 000 articles with 500 papers for testing and the remaining as training. NUS has only 211 articles all for testing. Krapivin contains 2, 304 articles with 400 papers as testing datasets and the others as training datasets. SemEval has 288 papers and it uses 100 articles for testing, the rest for training.

### 4.2. Experiment setup

Our experiment is separated into two parts, i.e., generating the present keyphrases and extracting absent keyphrases respectively.

As for the data pre-process, we transfer the training datasets into 2, 780, 316 pairs of (source text, keyphrase), in which the source texts are composed of the title and abstract of those publications, and keyphrases are the keywords assigned by the authors. We also lowercase the source texts, and apply the tokenization and transfer the digits into <digit> as symbols. As for the vocabulary constructed by LVT, we choose 50, 000 words with the highest frequency. We conduct two models, one is the basic seq2seq RNN model as the baseline,denoted as RNN, the other is the seq2seq RNN model equipped with the switching generator-pointer, denoted as SwitchRNN.

As for the parameters, the dimension of the hidden states in encoder and decoder, and the input and output embeddings are all set to be 256. The word embeddings are directly generated by word2vec, but it can be adapted during training. We use stochastic gradient descent to train our model with the initial learning rate set as 0.1. We also adopt an early stopping mechanism when the output results on validation set converges to save the training time. We set the training batch as 64 texts and the dropout rate is set as 0.5.

To evaluate our model between other baselines, for generating the present keyphrases, the evaluation metric we adopt is the F-measure, more specifically, F1 performance of top 5 and top 10 generated keyphrases denoted as F1@5 and F1@10. For generating the absent keyphrases, we choose the metric recall to evaluate our model. In specific, we evaluate the top 10 and top 50 recall results denoted as R@10 and R@50.

The baselines we compare include supervised and unsupervised methods. For supervised models, we choose Maui [13] and KEA [22]. For unsupervised models, we have four alternatives namely TextRank [16], Tf-Idf [4], ExpandRank [21] and SingleRank [21]. For their parameters, we directly use the best setups reported in their own papers.

# 5. Experimental Results

This section we introduce our experimental results along with the analysis.

Table 2: Predicting Present Keyphrases

| Datasets | Inspec | | Krapivin | | NUS | | SemEval | | KP20k | |
|---|---|---|---|---|---|---|---|---|---|---|
| Metric | F1@5 | F1@10 | F1@5 | F1@10 | F1@5 | F1@10 | F1@5 | F1@10 | F1@5 | F1@10 |
| Tf-Idf | 0.219 | 0.311 | 0.128 | 0.161 | 0.134 | 0.183 | 0.127 | 0.192 | 0.101 | 0.129 |
| TextRank | 0.283 | 0.222 | 0.187 | 0.163 | 0.192 | 0.197 | 0.175 | 0.185 | 0.174 | 0.146 |
| SingleRank | 0.212 | 0.305 | 0.186 | 0.162 | 0.138 | 0.170 | 0.133 | 0.175 | 0.094 | 0.118 |
| ExpandRank | 0.207 | 0.302 | 0.076 | 0.124 | 0.129 | 0.163 | 0.137 | 0.168 | N/A | N/A |
| Maui | 0.041 | 0.043 | 0.251 | 0.217 | 0.248 | 0.265 | 0.043 | 0.038 | 0.269 | 0.228 |
| KEA | 0.097 | 0.125 | 0.112 | 0.154 | 0.068 | 0.083 | 0.024 | 0.025 | 0.170 | 0.152 |
| RNN | 0.261 | 0.322 | 0.278 | 0.231 | 0.257 | 0.284 | 0.191 | 0.227 | 0.284 | 0.241 |
| SwitchRNN | 0.279 | 0.343 | 0.311 | 0.267 | 0.333 | 0.325 | 0.216 | 0.284 | 0.331 | 0.265 |

Table 3: Extracting Absent Keyphrases of seq2seq Models

| Datasets | Inspec | | Krapivin | | NUS | | SemEval | | KP20k | |
|---|---|---|---|---|---|---|---|---|---|---|
| Metric | R@10 | R@50 | R@10 | R@50 | R@10 | R@50 | R@10 | R@50 | R@10 | R@50 |
| RNN | 0.261 | 0.322 | 0.278 | 0.231 | 0.257 | 0.284 | 0.191 | 0.227 | 0.284 | 0.241 |
| SwitchRNN | 0.279 | 0.343 | 0.311 | 0.267 | 0.333 | 0.325 | 0.216 | 0.284 | 0.331 | 0.265 |

## 5.1. Predicting present keyphrases

Since most previous models only consider generate or extract present keyphrases, so we first conduct a task that only focuses on generating present keyphrases so as to make a fair comparison. The experimental results of all the baselines and our SwitchRNN is illustrated in detail in Table II. Note that ExpandRank does not have results on Krapivin due to the high time complexity.

From the results we can observe that as those unsupervised models, Tf-IDF, TextRank, SingleRank and ExpandRank, they are kind of more robust among all datasets. While for those supervised model Maui and KEA, their results vary largely between different datasets. Specifically, Maui obtains best results (except from our models) on three datasets Krapivin, NUS and KP20K, while not that good on the other two datasets Inspec and SemEval.

As for our seq2seq model based on RNN, we can see that both RNN and SwitchRNN outperforms consistently and significantly than other baselines no matter supervised or unsupervised approaches. This is because that seq2seq model with RNN is more expressive than the other alternatives as it could capture the semantics of the source text more accurately. However, RNN is still uncomparable to SwitchRNN due to that it focuses only on the semantic of the source so as to generate the words that are not very pertinent to the source. Actually, we find that about 2.5% (70, 891/2, 780, 316) keyphrases contain the words that are absent from the LVT vocabulary. The SwitchRNN, in the contrast, considers the location and syntactic information of the origin text by the help of switching generator-pointer, thus obtaining the best results among all the other models.

### 5.2. Generating absent keyphrases

For those keyphrases consist of absent words from the source text, all the other models except seq2seq model are not able to generate them. So we only compare our RNN and SwitchRNN models on generating absent keyphrases.

Table III demonstrates the R@10 and R@50 results of these two models on different datasets. We can find that just like generating the present keyphrases, SwitchRNN outperforms RNN model consistently and significantly. This proves that our models are able to capture the semantics of the source text. However, the improvement is not that obvious for SwitchRNN which further verifies the importance of capturing the features of the source text.

## 6. Conclusion

In this work, we propose a new RNN seq2seq model to generate keyphrases from the source text. Unlike conventional extractive models, our model could better capture the semantics of the source with the help of seq2seq framework. In addition, as we utilize a LVT approach to generate keyphrases, it could help us improve the efficiency, meanwhile generate the words absent from the source, which is an ability that many previous researches unable to accomplish. We also incorporate a switching generator-pointer to help us extract the rare words from the source yet not included in the target vocabulary. We conduct extensive experiments on real-life datasets on predicting present and absent keyphrases, respectively. Our model performs a lot better than the other alternatives which further proves its effectiveness.

## 7. References


[1] G. Berend. Opinion expression mining by exploiting keyphrase extraction. In IJCNLP, pages 1162–1170, 2011.

[2] K. Cho, B. van Merrienboer, and et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In EMNLP, pages 1724–1734, 2014.

[3] F. A. Gers and J. Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. IEEE Transactions on Neural Networks, 12(6):1333–40, 2001.

[4] K. S. Hasan and V. Ng. Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art. In COLING, pages 365–373, 2010.

[5] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In EMNLP, pages 216–223, 2003.

[6] S. Jean, K. Cho, and et al. On using very large target vocabulary for neural machine translation. In ACL, pages 1–10, 2015.

[7] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 60(5):493–502, 2004.

[8] S. N. Kim, O. Medelyan, and et al. Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles.

In SemEva Workshop, pages 21–26, 2010.

[9] M. Krapivin, A. Autaeu, and M. Marchese. Large dataset for keyphrases extraction. Technical Report DISI-09-055, 2009.

[10] T. T. N. Le, M. L. Nguyen, and et al. Unsupervised keyphrase extraction: Introducing new kinds of words to keyphrases. In AI, pages 665–671, 2016.

[11] Z. Liu, X. Chen, and et al. Automatic keyphrase extraction by bridging vocabulary gap. In CoNLL 2011, pages 135–144, 2011.

[12] Z. Liu, P. Li, and et al. Clustering to find exemplar terms for keyphrase extraction. In EMNLP, pages 257–266, 2009.

[13] O. Medelyan, E. Frank, and et al. Human-competitive tagging using automatic keyphrase extraction. In EMNLP, pages 1318–1327, 2009.

[14] O. Medelyan, I. H. Witten, and et al. Topic indexing with wikipedia. AAAI Press, 2008.

[15] R. Meng, S. Zhao, and et al. Deep keyphrase generation. In ACL, pages 582–592, 2017.

[16] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. Unt Scholarly Works, pages 404–411, 2004.

[17] T. D. Nguyen and M. Kan. Keyphrase extraction in scientific publications. In ICADL, pages 317–326, 2007.

[18] M. Song, I. Song, and et al. Kpspotter: a flexible information gain-based keyphrase extraction system. In CIKM workshop, pages 50–53, 2003.

[19] J. Tang, J. Z. Li, and et al. Loss Minimization Based Keyword Distillation. Springer Berlin Heidelberg, 2004.

[20] T. Tomokiyo and M. Hurst. A language model approach to keyphrase extraction. In ACL 2003 Workshop, pages 33–40, 2003.

[21] X. Wan and J. Xiao. Single document keyphrase extraction using neighborhood knowledge. In AAAI, pages 855–860, 2008.

[22] I. H. Witten, G. W. Paynter, and et al. KEA: practical automatic keyphrase extraction. In 4th ACM conference on Digital Libraries.

[23] Q. Zhang, Y. Wang, and et al. Keyphrase extraction using deep recurrent neural networks on twitter. In EMNLP, pages 836–845, 2016.

[24] Y. Zhang, E. E. Milios, and et al. A comparative study on key phrase extraction methods in automatic web site summarization. JDIM, 5(5):323–332, 2007.

[25] Y. Zhang, A. N. Z.H., and et al. World wide web site summarization. Web Intelligence and Agent Systems, 2(1):39–53, 2004.