Deep Keyphrase Completion

Jia Song (36920221153111, AI class),[1] Hejuan Tan (36920221153115, AI class), [1] Zhibing Lan (30920221154254, AI class), [2] Hao Yue (30920221154260, AI class), [2]

[1]Institute of Artificial Intelligence, Xiamen University
[2] School of Informatics Xiamen University(National Demonstrative Software School)

Abstract

Keyphrase provides accurate information of document content that are highly compact, concise, full of meanings, and widely used for discourse comprehension, organization, and text retrieval. Though previous studies have made substantial efforts for automated keyphrase extraction and generation, surprisingly, few studies have been made for *keyphrase completion* (KPC). KPC aims to generate more keyphrases for document (e.g. scientific publication) taking advantage of document content along with a very limited number of known keyphrases, which can be applied to improve text indexing system, etc. In this paper, we propose a novel KPC method with an encoder-decoder framework. We name it *deep keyphrase completion* (DKPC) since it attempts to capture the deep semantic meaning of the document content together with known keyphrases via a deep learning framework. Specifically, the encoder and the decoder in DKPC play different roles to make full use of the known keyphrases. The former considers the keyphrase-guiding factors, which aggregates information of known keyphrases into context. On the contrary, the latter considers the keyphrase-inhibited factor to inhibit semantically repeated keyphrase generation. Extensive experiments on benchmark datasets demonstrate the efficacy of our proposed model.

Introduction

Keyphrases are highly concise phrases that can provide compact and accurate information of given document content. Since high-quality keyphrases can facilitate the understanding, organizing, and accessing of document content, they are wildly used for NLP tasks, such as document clustering (Hammouda, Matute, and Kamel 2005; Chiu et al. 2020), text summarization (Qazvinian, Radev, and Ozgur 2010; Cano and Bojar 2019), and text retrieval (Boudin, Gallina, and Aizawa 2020). Due to the broadly demand, many automatic keyphrases extraction and generation methods are proposed over the years (Meng et al. 2017).

In this paper, we concentrate on the problem of ***keyphrase completion*** **(KPC)**, which aims to predict a relative large fixed size of keyphrases for document management. Consider the following situation where a publications retrieval system is going to be established and improved, each academic paper is demanded for *N* keyphrases to fill in while

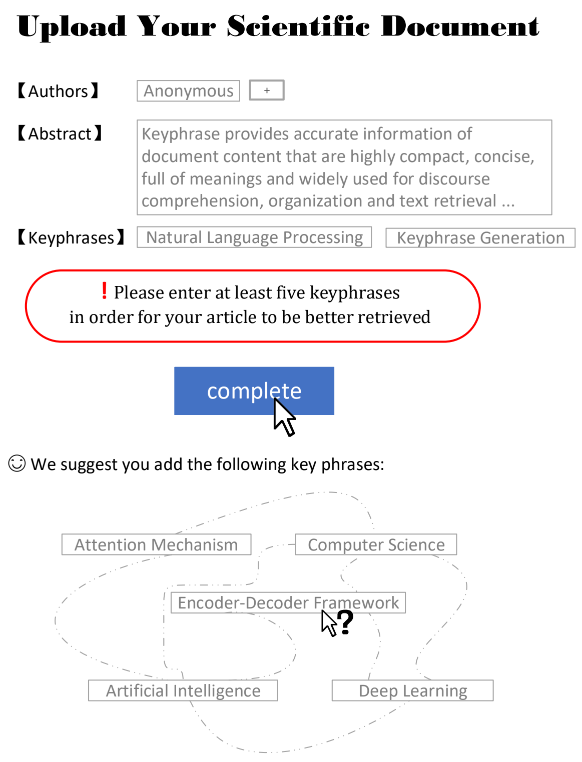

Upload Your Scientific Document
【Authors】
Anonymous
+
【Abstract】
Keyphrase provides accurate information of document content that are highly compact, concise, full of meanings and widely used for discourse comprehension, organization and text retrieval ...
【Keyphrases】
Natural Language Processing
Keyphrase Generation
! Please enter at least five keyphrases in order for your article to be better retrieved
complete
☺ We suggest you add the following key phrases:
Attention Mechanism
Computer Science
Encoder-Decoder Framework
?
Artificial Intelligence
Deep Learning

Figure 1: Example of keyphrase completion for text retrieval system.

it only provides a very limited number of known ones (i.e. much lower than *N*) and cannot meet the demand. In this kind of cases, it needs and expects for KPC, i.e. to predict more keyphrases for completion.

Most existing keyphrase prediction models, whether they be extracted-based methods (Mihalcea and Tarau 2004; Wan and Xiao 2008) or generated-enhanced methods (Meng et al. 2017), solely take advantage of the content of document. In fact, as we mentioned before, a document like scientific publication usually has already provided a few known keyphrases, with indicative topic information for the paper.

For example, in Figure 1 we show an example of keyphrase completion for text retrieval system, in which the scientific document have already existed two known keyphrases and be demanded at least five ones to complete. A serious drawback of using existing models for KPC is that they ignore the role of the known keyphrases and consequently fail to consider the already summarized information about them.

In this paper, we propose a novel KPC method with an encoder-decoder framework. We name it as *Deep KeyPhrase Completion* (DKPC) since it attempts to capture the deep semantic meaning of the document content together with known keyphrases via a deep learning framework.

Specifically, we believe that the known keyphrases have different roles in keyphrase completion, including keyphrase-guiding factors and keyphrase-inhibited factors. Firstly, we take the keyphrase-guiding factors into consideration for the encoder, which aggregates information of known keyphrases into context. Secondly, we take the keyphrase-inhibited factor into consideration for the decoder, which inhibits semantically repeated keyphrase generation. We conduct a comprehensive comparison on four benchmark datasets against two unsupervised models and two supervised deep learning models as baselines, and the results demonstrate the effectiveness of our proposed model.

The contributions of this paper are threefold:

- To the best of our knowledge, this is the first work that considers the problem of keyphrase completion using the deep learning method.
- We propose a novel KPC method for keyphrase completion using an encoder-decoder framework.
- We conduct a comprehensive comparison on four benchmark datasets against four baselines, and the results demonstrate the effectiveness of our proposed model.

Related Work

Different from keyphrase extraction, keyphrase generation models could generate absent keyphrases by modeling such task as a sequence-to-sequence (seq2seq) learning problem. Meng et al. (2017) first proposed CopyRNN, a seq2seq framework with copy mechanism for keyphrase generation. Following this framework, many enhanced variants of CopyRNN are proposed (Ye and Wang 2018; Chen et al. 2019; Cano and Bojar 2019; Chan et al. 2019; Yuan et al. 2020; Chen et al. 2020; Liu, Lin, and Wang 2020). For example, Ye and Wang (2018) proposed a semi-supervised methods by leveraging both labeled and unlabeled data. Chen et al. (2019) proposed TG-Net taking title information into consideration. Chan et al. (2019) utilized reinforcement learning with adaptive rewards to generate more sufficient and accurate keyphrase. Chen et al. (2020) designed a hierarchical decoding process and an exclusion mechanism to avoid generating duplicated keyphrases. Moreover, Swaminathan et al. (2020a,b) proposed a keyphrase generation approach using conditional Generative Adversarial Networks. (Ye et al. 2021) proposed a new training paradigm ONE2SET without predefining an order to concatenate the keyphrases. (Ahmad et al. 2021) proposed a method for neural keyphrase generation with layer-wise coverage attention.

Methodology

In this section, we introduce the details of our method. As we stated before, the existing keyphrases play two kinds of roles in keyphrase completion, as follows:

- ***Guiding Factor*** Since keyphrases are a group of words that condense the core information of a document, they always belong to the related topics. For instance, the keyphrases of this paper, "keyphrase generation" and "attention mechanism" are both related to Natural Language Processing. Motivated by this observation, we can capture such relevance and let existing ground-truth keyphrases to guide us complete more keyphrases. We call it the guiding factor.
- ***Inhibiting Factor*** It is reported that the keyphrases of more than 85% documents in the largest keyphrase generation benchmark dataset have different first word (Chen et al. 2020). In the generation stage, if the first word of the predicted keyphrase is the same as one of the existing keyphrases, it will be eliminated. We call it inhibiting factor.

Therefore, motivated the basic ideas above, our proposed model contains two modules: keyphrase-guided encoder and keyphrase-inhibited decoder. Before we introduce our model, we first give a formal problem definition.

Problem Definition

The task of keyphrase generation is usually formulated as follows: given a text dataset $\mathcal{D} = \{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^N$ where $\mathbf{x}_i$ is i-th source text and $\mathbf{y}^i = \{\mathbf{y}^{i,1}, \mathbf{y}^{i,2}, ..., \mathbf{y}^{i,M}\}$ is a set of keyphrases of it. N is the number of documents, and M is the number of keyphrases of the i-th document. Both the $\mathbf{x}_i$ and the j-th keyphrase of it are sequences of words, denoted as $\mathbf{x}^i = ({}^i_1, x^i_2, ..., x^i_{L_{x^i}})$ and $\mathbf{y}^{i,j} = (y^{i,j}_1, y^{i,j}_2, ..., y^{i,j}_{L_{y^{i,j}}})$, where L_{x^i} and $L_{y^{i,j}}$ are the number of words of the i-th texts and its j-th keyphrase respectively.

The goal of the model is to map from $\mathbf{x}$ to $\mathbf{y}$. Since our task is complete more keyphrases based on document content and a limited number of known keyphrases, we split the data into $(\mathbf{x}, \mathbf{k}, \mathbf{y})$ where $\mathbf{x}$ is source text, $\mathbf{k}$ is the given existing keyphrases of document x in advance, and $\mathbf{y}$ is the rest true keyphrases that we expect to generate for document completion.

Keyphrase-Guided Encoder

In the keyphrase-guided encoder, we aim to utilize the known keyphrases to guide more keyphrase generation. To this end, we take advantage of the promising attention mechanism in the encoder, which can gather the information of given keyphrases to the source text.

Source Text Representation. For words in source text, we use a bi-directional GRU Cho et al. (2014) to learn the contextual representation. The word embeddings are obtained from an initial embedding lookup table. The encoder Bi-GRU reads the input sequence $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{L_x}})$ forwardly and backwardly to converts them into two sets of hidden representation by iterating with the following equation along time t, respectively.

$$\overrightarrow{\mathbf{h}_t^x} = f(\mathbf{x}_t, \overrightarrow{\mathbf{h}_{t-1}^x}) \ , \tag{1}$$

$$\overleftarrow{\mathbf{h}_t^x} = f(\mathbf{x}_t, \overleftarrow{\mathbf{h}_{t-1}^x}) \ , \tag{2}$$

where f is a non-linear function. $\overrightarrow{\mathbf{h}_t^x}$ is the hidden state at time t when input sequence is ordered from $\mathbf{x}_1$ to $\mathbf{x}_{L_x}$ and $\overleftarrow{\mathbf{h}_t^x}$ is the same when input sequence is ordered from $\mathbf{x}_{L_x}$ to $\mathbf{x}_1$. L_x is the number of words of source text $\mathbf{x}$. The forward hidden state and backward hidden state are concatenated as $\mathbf{h}_i^x = [\overrightarrow{\mathbf{h}_i^x}; \overleftarrow{\mathbf{h}_i^x}]$ to represent the i-th word of $\mathbf{x}$.

Keyphrase Representation. Simple averaging is sufficient for keyphrase representation since keyphrase typically consists of a small number of words (in most cases 1 or 2). On the contrary, complicated models (like CNN or RNN) tend to overfit (Xin et al. 2018). Therefore, similar to (Xin et al. 2018), if a given keyphrase contains n_k words, we use the average of embedding vectors of the n_k words to represent this keyphrase, as follows:

$$\mathbf{k} = \frac{\mathbf{w}_1 + \mathbf{w}_2 + ... + \mathbf{w}_{n_k}}{n_k} \ , \tag{3}$$

where $\mathbf{k}$ is an aggregated semantic representation vector and w_i is the embedding vector of i-th word of this keyphrase from the embedding lookup table. In the phrase of model training, these vectors will be constantly updated to learn more appropriate semantic representations.

Attentional Hidden State. To effectively exploit the information provided by given keyphrases, an attention layer is used to calculate the relevance of given keyphrases and source text, and then obtain an aggregated information vector. This formal process is as follows:

$$e_{ij} = (\mathbf{h}_i^x)^\top \mathbf{W_1} \mathbf{k_j} \ , \tag{4}$$

$$\alpha_{i,j} = \frac{exp(e_{ij})}{\sum_{n=1}^{n_K} exp(e_{in})} \ , \tag{5}$$

$$\mathbf{c}_i = \sum_{j=1}^{n_k} \alpha_{i,j} \mathbf{k}_j \ , \tag{6}$$

where α_{ij} is the normalized attention score of the i-th word of source text and the j-th given keyphrase. $\mathbf{c}$ is an aggregated information vector which summarizes the information of all the given keyphrases. It is calculated based on the similarity of two vectors. We use the general mode as our alignment function Luong, Pham, and Manning (2015).

Finally, we concatenate the aggregated information vector of given keyphrases $\mathbf{c}$ and the source text representation $\mathbf{h}^x$ as encoder hidden state $\mathbf{h}$ for the downstream decoding process.

$$\mathbf{h} = \ [\mathbf{c} \| \mathbf{h}^x \ ] \tag{7}$$

Therefore, we can obtain $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{L_x}]$ according to this formula at different time steps. With the proposed keyphrase-guided encoder, the existing keyphrases can provide some useful information related to the domain of the source text and then guide to complete more ground-truth keyphrases.

Keyphrase-Inhibited Decoder

In the keyphrase-inhibited decoder, we aim to inhibit semantically repeated keyphrase generation. To this end, we employ Bi-GRU with attention mechanism and copy mechanism (Gu et al. 2016) as the building blocks of the decoder. It decompresses the source text into a context vector through an attention layer and generates the target keyphrase word by word. The process of Decoder is designed as follows:

$$\mathbf{s}_t = \text{GRU}([\mathbf{e}_{t-1}; \zeta_{t-1}], \mathbf{s}_{t-1}) \ , \tag{8}$$

$$\mathbf{c}_t = attn(\mathbf{s}_t, [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{L_x}], \mathbf{W_2}) \ , \tag{9}$$

$$\widetilde{\mathbf{h}_t} = tanh(\mathbf{W_3}[\mathbf{c_t}; \mathbf{s_t}]) \ , \tag{10}$$

where $\mathbf{e}_{t-1}$ is the embedding vector of y_{t-1} and ζ_{t-1} is position representation of it by selective reading in copy mechanism. The context vector $\mathbf{c}_t$ is the aggregated vector for $\mathbf{s}_t$ from the encoder's source text representation $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{L_x}]$ via attention mechanism.

Target hidden state $\mathbf{s}_t$ and the context vector $\mathbf{c}_t$ that weighs the sum of the hidden state of the encoder are combined through a simple concatenation layer to produce an attentional hidden sate. Later, the attentional hidden state goes through a softmax layer and outputs the probability distribution. Note that the probability of a word be produced is composed of two parts: the probability of generating it and the probability of coping it from source text. The process is as follows:

$$p(y_t|y_{t-1}, \mathbf{x}, \mathbf{k}) = p(y_t, g|y_{t-1}, \mathbf{x}, \mathbf{k}) + p(y_t, c|y_{t-1}, \mathbf{x}, \mathbf{k}) \ , \tag{11}$$

where $p(y_t, c|)$ is the probability of y_t being copied from source text and $p(y_t, g|)$ is the probability of y_t being generating from the large vocab. Specifically, the calculation process is as follows:

$$p(y_t, g|) = softmax(\mathbf{W_4}\widetilde{\mathbf{h}_t}) \ , \tag{12}$$
$$p(y_t, c|) = tanh(\mathbf{W_5}\mathbf{h}_t)\mathbf{s}_t \ . \tag{13}$$

Finally, normalize the sum of the two probabilities as the final probability that y_t is generated at the current step. If y_t is the same as the initial word of one of the given keyphrases, then it won't be the output as result.

Training

The widely used negative log likelihood loss is adopted to train our model:

$$\mathcal{L} = -\sum_{t=1}^{n} \log p(y_t|\mathbf{y_{t-1}}, \mathbf{x}, \mathbf{k}) \ , \tag{14}$$

where $\mathbf{y}_t$ is the t-th predicted word sequence. n is the length of target keyphrase $\mathbf{y}_t$, y_t is the t-th word of which.

Experiments

Experimental Settings

Datasets We utilize the public benchmark datasets for the experiments, includes: *KP20k* (Meng et al. 2017), *Inspec* (Hulth 2003), *Krapivin* (Krapivin, Autaeu, and Marchese 2009), *SemEval 2010* (Kim et al. 2010).

- ***KP20k*** (Meng et al. 2017) is the largest dataset on Keyphrase Generation. *KP20k* contains 567,830 scientific articles, where 20,000 articles for training and 20,000 articles for testing.
- ***Inspec*** (Hulth 2003) is composed of 2000 abstracts of scientific articles totally. The testing part which is composed of 500 abstracts is used to test our model.
- ***Krapivin*** (Krapivin, Autaeu, and Marchese 2009) is not split to testing part and training part by its original authors. We thus follow the set in (Meng et al. 2017) that uses the first 400 papers in alphabetical order as the testing part in total 2,304 papers.
- ***SemEval 2010*** (Mahata et al. 2018) consists of 288 full length ACM articles. The testing part has 100 articles.

Note that our model is only trained once using the training of *KP20k*, and evaluated on the testing part of all other datasets, which is a normal strategy like in CopyRNN (Meng et al. 2017), TG-Net (Chen et al. 2019), and ParaNet (Zhao and Zhang 2019).

Baselines To demonstrate the effectiveness of our proposed model for keyphrase extraction, we compare it with the following baselines, including two unsupervised statistic-based methods TF-IDF, TextRank (Mihalcea and Tarau 2004) and deep learning models RNN (Meng et al. 2017), CopyRNN (Meng et al. 2017).

Evaluation Protocol We follow the common practice and evaluate the performance of our model in previous work (Meng et al. 2017). Similar to them, we utilize F-measure (F1)@N as an evaluation metric based on the macro-averaged Precision, Recall. N indicates the number we need to complete, which is set among {5, 10 or 50}. The precision is computed by the number of keyphrases generated correctly over the number of all generated keyphrases. The recall is calculated by the number of keyphrases generated correctly over the number of keyphrases that are the targets of the sample. For each method, we give the F-measure at top 5 and top 10 predictions on four real-world benchmark datasets.

Implementation Details During data preprocessing, we tokenize, lowercase, and stemming the text. After that, we use the symbol $< digit >$ to replace each digit. Since the task is to use some pre-designated keyphrases and document content to complete more useful keyphrases, we need to take out a few keyphrases before training. M denotes the number of original targets of a data record and N denotes the number of pre-designated keyphrases we need to take out at random. We design the processing rules as follows:

$$N = \begin{cases} delete\ this\ sample, & if\ M = 0 \\ 1, & if\ M \in \{2,3\} \\ 2, & if\ M \in \{4,5\} \\ 3, & if\ M \in [6,20] \\ 5, & if\ M > 20 \end{cases} \tag{15}$$

The vocabulary includes 50,000 words that appear most frequently in the datasets. We set the embedding dimension to 200 and the hidden size to 100. We use teacher forcing to help training model. One batch contains 128 data samples. Our model is optimized by Adam (Kingma and Ba 2014) with an initial learning rate 0.001. The learning rate will update every 10000 batches according to StepLR[1]. When F1@10 on the validation set doesn't rise for 100 consecutive steps, we stop training early. During testing phrase, we use beam search to select effective keyphrases and beam size is set to 50.

Results and Analysis

Table 1 shows the performance of our method against four baselines, from which we observe that our proposed model performs better than all baselines for keyphrase completion in terms of both F1@5 and F1@10 metrics on four datasets. The best scores are highlighted in bold. It confirms the capability of our method in modeling the deep semantic meaning of document content and known keyphrases for keyphrases completion.

Analysis To make a fair comparison, we take out a few keyphrases in advance as known keyphrases according to our designed rules mentioned before, which is applied to all testing datasets. In this case, fewer predictable keyphrases will aggravate the keyphrase prediction challenges for all methods, which leads to lower scores[2].

As mentioned before, the key motivation behind our work is that we are interested in the proposed model's capability for completing keyphrases based on document content and limited known keyphrases. It is worth noting that such a completion task is a challenging problem. To the best of our knowledge, no existing pertinent methods are proposed specifically to handle this new task. The unsupervised keyphrase extraction models, i.e. TF-IDF and TextRank, and the keyphrase generation models, RNN and CopyRNN, inevitably fail to take the known keyphrases into consideration. We can observe from Table 1 that all baseline perform worse than ours.

In KPC task, the known keyphrases can play an important role. We believe they are always highly semantically related to other unknown keyphrases that needed to be completed. Our method can capture such relation and utilize known semantic information to select the most conducive sentences to the keyphrases completion from the source document with an attention mechanism. In addition, the given keyphrases are also aggregated into the context representation which

[1]A method for interval adjustment of learning rate in PyTorch.

[2]The absolute scores in Table 1 are thus lower than their performance in original papers.

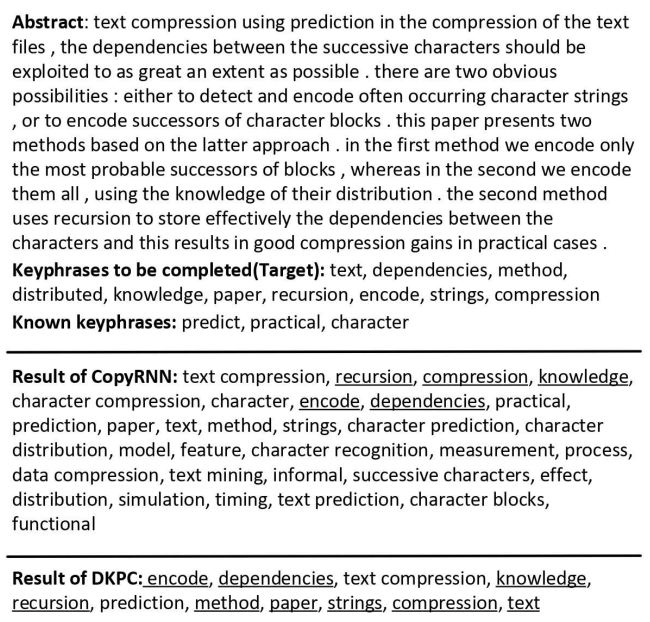
Table 1: The performance of expanding keyphrases of various models on four benchmark datasets.

Methods	**Inspec**		**Krapivin**		**SemEval**		**KP20k**	
	F1@5	**F1@10**	**F1@5**	**F1@10**	**F1@5**	**F1@10**	**F1@5**	**F1@10**
TF-IDF	0.107	0.151	0.068	0.067	0.061	0.073	0.108	0.101
TextRank	0.172	0.190	0.076	0.084	0.028	0.039	0.065	0.065
RNN	0.084	0.082	0.068	0.060	0.077	0.084	0.103	0.087
CopyRNN	0.179	0.194	0.127	0.106	0.106	0.128	0.161	0.142
DKPC (Ours)	**0.212**	**0.218**	**0.161**	**0.138**	**0.120**	**0.141**	**0.181**	**0.154**

Table 2: The performance of CopyRNN and our proposed model in the same case.

Methods	**F1@5**	**F1@10**	**P@5**	**P@10**	**R@5**	**R@10**	**Similarity**
CopyRNN	0.400	0.500	0.600	0.500	0.300	0.50	0.387
DKPC	**0.533**	**0.800**	**0.800**	**0.800**	**0.400**	**0.800**	**0.473**

more fully exploits the information already known. On the four datasets, our method achieves a cumulative 11.3% improvement at F1@5 and F1@10.



Abstract: text compression using prediction in the compression of the text files , the dependencies between the successive characters should be exploited to as great an extent as possible . there are two obvious possibilities : either to detect and encode often occurring character strings , or to encode successors of character blocks . this paper presents two methods based on the latter approach . in the first method we encode only the most probable successors of blocks , whereas in the second we encode them all , using the knowledge of their distribution . the second method uses recursion to store effectively the dependencies between the characters and this results in good compression gains in practical cases .

Keyphrases to be completed(Target): text, dependencies, method, distributed, knowledge, paper, recursion, encode, strings, compression

Known keyphrases: predict, practical, character

Result of CopyRNN: text compression, recursion, compression, knowledge, character compression, character, encode, dependencies, practical, prediction, paper, text, method, strings, character prediction, character distribution, model, feature, character recognition, measurement, process, data compression, text mining, informal, successive characters, effect, distribution, simulation, timing, text prediction, character blocks, functional

Result of DKPC: encode, dependencies, text compression, knowledge, recursion, prediction, method, paper, strings, compression, text

Figure 2: A keyphrase completion example of our proposed DKPC compare to previous method CopyRNN.

Case Study

Figure 2 and Table 2 show a case study of keyphrase completion by our method against CopyRNN. we can observe from Figure 2 that our model not only completes keyphrases accurately but also completes more targets with less computational resources. Table 2 shows the performance of the model on Precision, Recall and F1-measure, from which we observe that DKPC is significantly better than CopyRNN. Specifically, we use Bert (Devlin et al. 2019) to obtain the embedding of each keyphrase, and then calculate similarity by non-repeating pairs between keyphrases. The sum of similarity divided by the number of keyphrase pairs is similarity. Due to the guidance of known keyphrases, our proposed method can search for accurate keyphrases from a smaller semantic space than CopyRNN's. That's why our model can find more real keyphrases in less time. The keyphrases inhibition factor is also helpful. For example, in contrast to CopyRNN, our model never includes the known keyphrase "practical" as a target to complete, or any keyphrase beginning with the first word of any known keyphrase.

Conclusion and Future Work

In this paper, we concentrate on the new keyphrase completion task. To the best of our knowledge, this is the first work that considers the problem of keyphrase completion. To solve this problem, we propose a novel KPC method that takes an encoder-decoder framework. The encoder and the decoder in our method make full use of the guidance factor and inhibition factor of known keyphrases, respectively. The comprehensive experimental results demonstrate the effectiveness of our proposed model in the task of keyphrase completion. Some interesting future work may include that Our model only explore post-processing for inhibition of known keyphrases. One interesting direction is to design some specific layers on the deep learning model structure to make the inhibition of known keyphrases work better.

References

Ahmad, W. U.; Bai, X.; Lee, S.; and Chang, K.-W. 2021. Select, Extract and Generate: Neural Keyphrase Generation with Layer-wise Coverage Attention. In *Proceedings of ACL*, 1389–1404.

Boudin, F.; Gallina, Y.; and Aizawa, A. 2020. Keyphrase Generation for Scientific Document Retrieval. In *Proceedings of ACL*, 1118–1126.

Cano, E.; and Bojar, O. 2019. Keyphrase Generation: A Text Summarization Struggle. In *Proceedings of NAACL*, 666–672.

Chan, H.; Chen, W.; Wang, L.; and King, I. 2019. Neural Keyphrase Generation via Reinforcement Learning with Adaptive Rewards. In *Proceedings of ACL*, 2163–2174.

Chen, W.; Chan, H.; Li, P.; and King, I. 2020. Exclusive Hierarchical Decoding for Deep Keyphrase Generation. In *Proceedings of ACL*, 1095–1105.

Chen, W.; Gao, Y.; Zhang, J.; King, I.; and Lyu, M. R. 2019. Title-Guided Encoding for Keyphrase Generation. In *Proceedings of AAAI*, 6268–6275.

Chiu, B.; Sahu, S. K.; Thomas, D.; Sengupta, N.; and Mahdy, M. 2020. Autoencoding Keyword Correlation Graph for Document Clustering. In *Proceedings of ACL*, 3974–3981.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Doha, Qatar: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1631–1640. Berlin, Germany: Association for Computational Linguistics.

Hammouda, K. M.; Matute, D. N.; and Kamel, M. S. 2005. Corephrase: Keyphrase extraction for document clustering. In *International workshop on machine learning and data mining in pattern recognition*, 265–274.

Hulth, A. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP*, 216–223.

Kim, S. N.; Medelyan, O.; Kan, M.-Y.; and Baldwin, T. 2010. SemEval-2010 Task 5: Automatic Keyphrase Extraction from Scientific Articles. In *Proceedings of Workshop on Semantic Evaluation, ACL*, 21–26.

Kingma, D.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *Computer Science*.

Krapivin, M.; Autaeu, A.; and Marchese, M. 2009. Large Dataset for Keyphrases Extraction.

Liu, R.; Lin, Z.; and Wang, W. 2020. Keyphrase Prediction With Pre-trained Language Model. In *arXiv preprint arXiv:2004.10462*.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421. Lisbon, Portugal: Association for Computational Linguistics.

Mahata, D.; Kuriakose, J.; Shah, R. R.; and Zimmermann, R. 2018. Key2Vec: Automatic Ranked Keyphrase Extraction from Scientific Articles using Phrase Embeddings. In *Proceedings of NAACL*, 634–6392018.

Meng, R.; Zhao, S.; Han, S.; He, D.; Brusilovsky, P.; and Chi, Y. 2017. Deep keyphrase generation. In *Proceedings of ACL*, 582–592.

Mihalcea, R.; and Tarau, P. 2004. TextRank: Bringing order into text. In *Proceedding of EMNLP*, 404–411.

Qazvinian, V.; Radev, D. R.; and Ozgur, A. 2010. Citation summarization through keyphrase extraction. In *Proceedings of COLING*, 895–903.

Swaminathan, A.; Gupta, R. K.; Zhang, H.; Mahata, D.; Gosangi, R.; and Shah, R. R. 2020a. Keyphrase Generation for Scientific Articles Using GANs. In *Proceedings of AAAI*, 13931–13932.

Swaminathan, A.; Zhang, H.; Mahata, D.; Gosangi, R.; Shah, R. R.; and Stent, A. 2020b. A Preliminary Exploration of GANs for Keyphrase Generation. In *Proceedings of EMNLP*, 8021–8030.

Wan, X.; and Xiao, J. 2008. Single Document Keyphrase Extraction Using Neighborhood Knowledge. In *Proceedings of AAAI*, 855–860.

Xin, J.; Lin, Y.; Liu, Z.; and Sun, M. 2018. Improving Neural Fine-Grained Entity Typing with Knowledge Attention. In *Proceedings of AAAI*, 5997–6004.

Ye, H.; and Wang, L. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of EMNLP*, 4142–4153.

Ye, J.; Gui, T.; Luo, Y.; Xu, Y.; and Zhang, Q. 2021. One2Set: Generating Diverse Keyphrases as a Set. In *Proceedings of ACL*, 4598–4608.

Yuan, X.; Wang, T.; Meng, R.; Thaker, K.; Brusilovsky, P.; He, D.; and Trischler, A. 2020. One Size Does Not Fit All: Generating and Evaluating Variable Number of Keyphrases. In *Proceedings of ACL*, 7961–7975.

Zhao, J.; and Zhang, Y. 2019. Incorporating Linguistic Constraints into Keyphrase Generation. In *Proceedings of ACL*, 5224–5233.