Searching Effective Transformer for Seq2Seq Keyphrase Generation

Yige Xu, Yichao Luo, Yicheng Zhou, Zhengyan Li, Qi Zhang, Xipeng Qiu(✉), and Xuanjing Huang

School of Computer Science, Fudan University, Shanghai, China
{ygxu18,ycluo18,yczou18,lzy19,qz,xpqiu,xjhuang}@fudan.edu.cn

Abstract. Keyphrase Generation (KG) aims to generate a set of keyphrases to represent the topic information of a given document, which is a worthy task of Natural Language Processing (NLP). Recently, the Transformer structure with fully-connected self-attention blocks has been widely used in many NLP tasks due to its advantage of parallelism and global context modeling. However, in KG tasks, Transformer-based models can hardly beat the recurrent-based models. Our observations also confirm this phenomenon. Based on our observations, we state the *Information Sparsity Hypothesis* to explain why Transformer-based models perform poorly in KG tasks. In this paper, we conducted exhaustive experiments to confirm our hypothesis, and search for an effective Transformer model for keyphrase generation. Comprehensive experiments on multiple KG benchmarks showed that: (1) In KG tasks, uninformative content abounds in documents while salient information is diluted globally. (2) The vanilla Transformer equipped with a fully-connected self-attention mechanism may overlook the local context, leading to performance degradation. (3) We add constraints to the self-attention mechanism and introduce direction information to improve the vanilla Transformer model, which achieves state-of-the-art performance on KG benchmarks.

Keywords: Keyphrase Generation · Transformer · Seq2Seq

1 Introduction

Keyphrase Generation (KG) is a classic and challenging task in Natural Language Processing (NLP) that aims at predicting a set of keyphrases for the given document. As keyphrases contain the core idea of the document, it is useful in various downstream tasks such as information retrieval [13], document clustering [12], opinion mining [2,28], and text summarization [27]. In most cases, keyphrases can be found in the given document, which means it is a substring of the source text (aka *present keyphrases*). In other challenging cases, some

Y. Xu and Y. Luo—Contribute equally.

L. Wang et al. (Eds.): NLPCC 2021, LNAI 13029, pp. 86–97, 2021.
https://doi.org/10.1007/978-3-030-88483-3_7

Table 1. Sample document with keyphrase labels. In this example, there are some *present keyphrases* (in red color) and some *absent keyphrases* (in blue color).

Document: Rental software valuation in it investment decisions. The growth of application service providers (asps) is very rapid, leading to a number of options to organizations interested in developing new information technology services. ... Some of the more common capital budgeting models may not be appropriate in this volatile marketplace. However, option models allow for many of the quirks to be considered. ...
Keyphrase labels: (present keyphrases) application service providers; options; capital budgeting; (absent keyphrases) information technology investment; stochastic processes; risk analysis

keyphrases may not appear in the given document (aka *absent keyphrases*). An examplar document along with the reference keyphrases can be found in Table 1.

In recent years, Transformer [25] has become prevailing in various NLP tasks, such as machine translation [25], summarization [19], language modeling [6], and pre-trained models [7]. Compared to recurrent-based models, such as GRU [5] and LSTM [10], the Transformer module introduces fully-connected self-attention blocks to incorporate global contextual information and capture the long-range semantic dependency [18]. Meanwhile, Transformer has a better parallelism ability than RNN due to the module structure.

However, the vanilla Transformer has a poor performance in the KG task, and can hardly beat the approaches based on RNN [4,21]. We carefully tuned Transformer models on the KG task, and our observations have confirmed this conclusion. Since Transformer-based models have been proved their success in many other NLP tasks in recent years [18], it is worth exploring why Transformer performs poorly in the KG task.

Considering the example shown in Table 1, the keyphrase "*application service providers*" is mentioned in the sentence "*The growth of application service providers...*", which may not have a close relationship with the other sentences in the example document. The RNN captures features by modeling the text sequence [10], and the Transformer focus more on the global context modeling [25]. Based on the module architecture and previous observations, we state the **Information Sparsity Hypothesis**: *In KG tasks, uninformative content abounds in documents while salient information is diluted in the global context.*

For a vanilla Transformer encoder, every two tokens from different segments can be attended by each other, while it may probably bring uninformative content to each other. Based on the property of Transformer and our hypothesis, we introduce two adaptations towards the objective Transformer encoder.

The first adaptation is to reduce the context that can be attended by the self-attention mechanism. To analyze the influence of the attention mechanism, we firstly manually separate the input sequence into several segments while tokens from different segments cannot be attended by each other. Experiments show that this modification leads to a slight improvement. Thus, we further add some constraints on the attention mechanism by sparsing the attention mask matrix.

The second adaptation is introducing direction information. Typically, previous models usually use BiRNN as encoder and decoder, distinguishing which side the context information comes from. But the vanilla Transformer only uses

a position embedding to distinguish tokens from different positions, which does not significantly contain direction information. Thus, Transformer-based model is usually unaware of the distance between different tokens. However, it is more important to model the relationship between tokens and their neighbors in the KG task, which mainly decides they should be or should not be part of the keyphrase. Following [24] and [6], we employ relative multi-head attention to the KG task. Our experiment shows that relative multi-head attention can bring improvement than multi-head attention.

Our main observations are summarized in: (1) The most informative content usually comes from its neighborhood tokens, which provide empirical evidence to our proposed *Information Sparsity Hypothesis*. (2) Due to this phenomenon, we confirm that Transformer performs poorly in KG tasks because the fully-connected self-attention mechanism on vanilla may overlook the local context. (3) We adapt Transformer module by reducing attention to uninformative content and adding direction information. Our proposed model achieves state-of-the-art results on present keyphrase generation.

2 Methodology

2.1 Reduce Attention to Uninformative Content

As mentioned in Sect. 1, we try to use Information Sparsity Hypothesis to explain why Transformer does not work on KG tasks. In RNN models, the attention mechanism mainly focuses on target keyphrase tokens. In Transformer models, the attention distribution is more sparse and tend to receive more information from different tokens. Hence, reducing attention to uninformative content is a considerable solution. In this section, we will introduce two methods to constraint the attention mechanism: mandatory prohibition, and soft prohibition.

Segmentation. Considering a an input sequence $\mathbf{x} = [x_1, x_2, \ldots, x_{l_\mathbf{x}}]$, token x_1 and $x_{l_\mathbf{x}}$ can be attended by each other due to the self-attention mechanism. For convenience, firstly we manually separate the input sequence $\mathbf{x}$ into several segments, for example, $\mathbf{x} = [\hat{x}_1, \ldots, \hat{x}_K]$:

$$\hat{x}_i = [x_{N\cdot(i-1)+1}, x_{N\cdot(i-1)+2}, \ldots, x_{N\cdot i}], \tag{1}$$

$$\hat{\mathbf{h}}_i = \text{Transformer}(\hat{x}_i), \tag{2}$$

$$\mathbf{H} = \text{concat}[\hat{\mathbf{h}}_1; \hat{\mathbf{h}}_2; \ldots; \hat{\mathbf{h}}_K], \tag{3}$$

where K denotes the number of segments, and N denotes the sequence length of each segment. The attention between tokens from different segments are prohibit.

Sparsing the Matrix of Attention Mask. In addition to the approaches mentioned above, another efficient way to reduce attention to uninformative content is by adding constraints to the attention mask. If the input sequence $\mathbf{x}$ has a sequence length of n, the attention mask matrix $\mathbf{M} \in \{0, 1\}^{n \times n}$ is:

$$\mathbf{M}_{i,j} = \begin{cases} 0, \text{ if } x_i \text{ does not attends } x_j \\ 1, \text{ if } x_i \text{ attends } x_j \end{cases}, \tag{4}$$

In general, the attention mechanism is fully-connected, which means any two tokens x_i and x_j can be attended by each other. Thus, the attention mask in the vanilla Transformer encoder is a matrix filled with one: $\mathbf{M}^{full} = 1^{n \times n}$.

However, based on our Information Sparsity Hypothesis, a given token x_i does not need to attend the uninformative tokens. Typically, the informative tokens usually come from three parts, including lead tokens that contain the core topic of the whole document, neighborhood tokens that contain the contextual information, and tokens with high attention scores that is highly revalent to x_i.

For attending lead tokens, the lead g tokens are selected to the interaction, so that the attention mask matrix can be formula as:

$$\mathbf{M}^{lead}_{i,j} = \begin{cases} 1, \text{ if } i \leq g \text{ or } j \leq g \\ 0, \text{ if } i > g \text{ and } j > g \end{cases}, \tag{5}$$

For neighborhood attention, each token x_i can mostly attend w tokens, which includes its previous $\lfloor \frac{w}{2} \rfloor$ tokens and its next $\lfloor \frac{w}{2} \rfloor$ tokens:

$$\mathbf{M}^{neigh}_{i,j} = \begin{cases} 1, \text{ if } |i-j| \leq \lfloor \frac{w}{2} \rfloor \\ 0, \text{ if } |i-j| > \lfloor \frac{w}{2} \rfloor \end{cases}, \tag{6}$$

For highly relationship attention, we firstly compute the attention score matrix and then select top k tokens to attend:

$$\mathbf{M}^{topk}_{i} \in [0,1]^n, \text{ where } \sum_{j=1}^{n} \mathbf{M}^{topk}_{i,j} = k \tag{7}$$

Similar to BigBird [32], the attention mask matrices can be mixed up by:

$$\bar{\mathbf{M}}_{i,j} = (\alpha \mathbf{M}^{lead}_{i,j}) \circ (\beta \mathbf{M}^{neigh}_{i,j}) \circ (\gamma \mathbf{M}^{topk}_{i,j}). \tag{8}$$

where α, β, γ are hyperparameters in $\{0, 1\}$ to control which attention mask will be used and which will not, and $\circ$ means element-wised "or" operation.

2.2 Relative Multi-head Attention

Inspired by the success of reducing attention to uninformative content, we further consider what else properties the Transformer lacks compared to RNN-based models. One empirical observation is that Transformer is not sensitive to the direction information because Transformer is not easy to distinguish whether the context comes from the left side or the right side. Hence, following the successful experience in Named Entity Recognition (NER) task [30], we introduce relative multi-head attention to improve our model. The relative multi-head attention

Table 2. Summary statistics of four scientific article benchmark.

Dataset	#Train	#Validation	#Test	#Avg. present	#Avg. absent
Inspec	–	1,500	500	7.64	2.10
Krapivin	–	1,903	400	3.27	2.57
SemEval	–	144	100	6.28	8.12
KP20k	509,818	20,000	20,000	3.32	1.93

uses a relative positional encoder $\mathbf{R}_{i-j}$ to replace the position encoding $\mathbf{P}_j$ in the vanilla self-attention mechanism [6,30], which can be formula as:

$$\begin{aligned}\mathbf{A}_{i,j}^{abs} = \mathbf{Q}_i\mathbf{K}_j^T = \mathbf{H}_i\mathbf{W}_q(\mathbf{H}_j\mathbf{W}_k)^T + \mathbf{H}_i\mathbf{W}_q(\mathbf{R}_{i-j}\mathbf{W}_k)^T \\ + \mathbf{u}(\mathbf{H}_j\mathbf{W}_k)^T + \mathbf{v}(\mathbf{R}_{i-j}\mathbf{W}_k)^T.\end{aligned} \tag{9}$$

where $\mathbf{R}$ is a not learnable sinusoid encoding matrix [25], $\mathbf{u}$ and $\mathbf{v}$ are two learnable parameters used to substitute the position-based query terms.

3 Experiment Settings

3.1 Notations and Problem Definition

In this paper, we use bold lowercase and uppercase letters to denote vectors and matrices, respectively. We use calligraphy letters to indicate the sets and $\mathbf{W}$ to represent a parameter matrix. Given an input document $\mathbf{x}$, the KG task aims to generate a set of ground-truth keyphrases $\mathcal{Y} = \{\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^{|\mathcal{Y}|}\}$, where $|\mathcal{Y}|$ indicates the keyphrase number of $\mathbf{x}$. Both the source document $\mathbf{x} = [x_1, \ldots, x_{l_\mathbf{x}}]$ and each target keyphrase $\mathbf{y}^i = [y_1^i, \ldots, y_{l_{\mathbf{y}_i}}^i]$ are word sequences, where $l_\mathbf{x}$ and $l_{\mathbf{y}_i}$ indicates the word numbers of $\mathbf{x}$ and $\mathbf{y}^i$ respectively.

3.2 Datasets

To verify and analyze the information sparsity hypothesis, we conduct experiments on four public datasets: **Inspec** [11], **Krapivin** [16], **SemEval-2010** [14], **KP20k** [21]. Documents from these four benchmarks are all scientific articles. In Table 2, we describe the detailed statistics of each dataset. Following [4,31], an article with title and abstract are included as the source data and a set of Keyphrases are included as the target data.

3.3 Evaluation Metrics

We use $F_1@5$ and $F_1@M$ as evaluation metric, which is the same as [4]. As for $F_1@5$, we cut off the top 5 keyphrases for calculating. When the number of predicted keyphrases is less than 5, we randomly append incorrect keyphrases until

it obtains 5 keyphrases. Unlike $F_1@5$ that using a fixed number for predictions, $F_1@M$ [31] compares all predictions with target keyphrases. Therefore, the effect of the evaluation metric caused by the different number of predictions should be considered. In this paper, we use the macro $F_1@M$ and $F_1@5$ scores to report results. Before calculating the scores, we should stem all keyphrases and remove all duplicated keyphrases.

3.4 Implementation Details

Following previous work [4,21,31], for data preprocessing, we lowercase the characters, tokenize the sequences and replace the digits into "⟨digit⟩" token. For the training stage, we use the source code from [4]. When training, we sort the present keyphrase targets in the order of their first occurrences. The absent keyphrase targets are then appended at the end of ordered present targets. We use a preprocessed vocabulary with 50,000 tokens, which is shared between the encoder and decoder. For RNN models, the dimension of encoder and decoder are kept as 300, and we use a 2-layer encoder and 1-layer decoder. For Transformer models, we carefully tuned the hyperparameters. All learnable parameters are randomly initialized by a uniform distribution in $[-0.1, 0.1]$ before the training stage. Due to the average length of about 180 tokens, it requires high GPU memory in Transformer models. Thus we set the batch size as 16 or 24. Following [4], we set the max gradient norm as 1.0 and the initial learning rate as $1e-3$ for RNN models and $1e-4$ for Transformer models, and remove the dropout. During the training stage, we use Adam as the optimizer. We save checkpoints every 5,000 steps for evaluation.

4 Results and Discussions

4.1 Applying Transformer to Keyphrase Generation

Table 3. Present keyphrase prediction results of different standard encoders.

Model	Inspec		Krapivin		SemEval		KP20k	
	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$
ExHiRD-h [4]	0.291	0.253	0.347	0.286	0.335	0.284	0.374	0.311
ExHiRD-h (Our implementation)								
w/BiGRU encoder	0.288	0.248	0.344	0.281	0.326	0.274	0.374	0.311
w/BiLSTM encoder	0.285	0.245	0.346	0.278	0.328	0.278	0.374	0.311
w/CNN encoder	0.284	0.247	0.347	0.277	0.324	0.269	0.371	0.306
w/Transformer encoder	0.267	0.240	0.325	0.271	0.304	0.258	0.355	0.299

Typically, previous model applied BiGRU on both encoder and decoders. Thus, we firstly replace the BiGRU encoder with other widely-used components: BiLSTM, CNN, and Transformer. The results are shown in Table 3. Our observation

confirms that Transformer encoder can hardly beat the recurrent-based models, while CNN encoder has comparable performance.

In this experiment, we note that the accurate of absent keyphrase generation is very poor. On KP20k, only about 1,000 absent keyphrases are correctly predicted, and the ground truth contains 38,531 keyphrases. The score on absent keyphrase prediction is not only inadequate but also in a high variance. We hypothesis that there are two main reasons: (1) the average number of keyphrases in each document is limited (mainly about 3 keyphrases), thus it is not easy to predict the correct results; (2) the evaluation metric is strict that only the accurate predictions are counted. Due to this reason, we mainly focus on present keyphrase generation in the following experiments.

4.2 Tuning Transformer Model

As shown in Sect. 4.1 and [4], Transformer performs poorly in the KG tasks. However, tuning Transformer with different hyperparameters affects the performance. Therefore, in this section, we will carefully fine-tune the Transformer model with different hyperparameters.

Table 4. Present keyphrase prediction results of different encoder-decoder pairs on KP20k. "$\#C@M$" and "$\#C@5$" indicates the number of correctly predictions in $F_1@M$ and $F_1@5$, respectively. "#Avg. Len" is the average number of predictions to present keyphrases. "TF" is standard Transformer module.

Encoder	Decoder	$F_1@M$	$F_1@5$	$\#C@M$	$\#C@5$	#Avg. Len
BiGRU	BiGRU	0.374	0.311	25,275	24,435	3.88
BiGRU	TF	0.372	0.298	24,252	23,603	3.84
TF	BiGRU	0.363	0.290	23,337	22,933	3.81
TF	TF	0.359	0.290	22,912	23,314	3.69

Effective of Different Encoder-Decoder Pairs. As shown in Table 4, we firstly explore the effectiveness of different encoder-decoder pairs in this section. According to our experiment, the $F_1@M$ score drops significantly if we use a standard Transformer encoder to replace the BiGRU encoder. Meanwhile, our experiment shows that the $F_1@M$ score will also obtain a slight change when we use a standard Transformer decoder to replace the BiGRU decoder.

In our experiments, we notice that the average length of target present keyphrases is 3.32, but models usually predict more than 3.5 keyphrases on average. We will randomly append keyphrases until there are five predicted keyphrases when we compute the $F_1@5$ score. Due to this reason, predicting more keyphrases tend to improve the $F_1@5$ score because a maybe-incorrect prediction is better than an absolutely-incorrect prediction. In contrast, when computing the $F_1@M$ score, a maybe-incorrect keyphrase is worse than predicting nothing because the overall accuracy is at a low level from about 0.3 to about

0.4. The average number of predicted keyphrases is also an important indirect factor for evaluating the ability of keyphrase generation. Hence, we will have a joint consideration between the $F_1@5$ score and the $F_1@M$ score for comparison in the following experiments.

Table 5. Present keyphrase results of different hyperparameters on KP20k.

$<\eta, N, A, H>$	$F_1@M$	$F_1@5$	#Avg. Len
$<1e-3, 2, 6, 300>$	0.348	0.278	3.57
$<1e-4, 2, 6, 300>$	0.359	0.290	3.69
$<1e-4, 3, 6, 300>$	0.359	0.284	3.86
$<1e-4, 3, 8, 512>$	0.362	0.299	3.89
$<1e-4, 4, 6, 300>$	0.363	0.296	3.83
$<1e-4, 4, 8, 512>$	0.364	0.300	3.96
$<1e-4, 6, 12, 768>$	0.364	0.304	4.15
$<1e-4, 12, 12, 768>$	0.361	0.293	3.77

Tuning Transformer Model with Different Hyperparameters. As mentioned in Sect. 4.2, Transformer encoder performs poorly in KP20k with general hyperparameters. Thus, we will carefully tune the Transformer models with different hyperparameters. The results are shown in Table 5. We mainly tuned four hyperparameters: learning rate η, number of layers N, number of attention heads A, and hidden size H. Though the hidden size of 300 is usually set in RNN-based models, it is not suitable in Transformer-based models. Meanwhile, more attention heads and layers are required. However, when we stack more layers (e.g., $N = 12$) into our model, we find that: (1) The training stage has become more difficult that obtains a slower convergence; (2) The validation score as well as test evaluation score are lower than a 6-layer or a 4-layer Transformer model. Thus, we set $N = 4$ or $N = 6$ in the following experiments.

4.3 Adapting Transformer to Keyphrase Generation

As shown in Sect. 4.2, we confirm that Transformer performs poorly in KG tasks. Therefore, we apply the methods shown in Sect. 2.

Reduce Attention to Uninformative Content. As shown in Table 6, chunking brings slight improvement on the KG task. It is worth noting that $N = 250$ and $N = 500$ have a similar result to not applying the chunking because the average length of the training set is about 180 tokens.

Table 6. Effects of segment length N on present keyphrase prediction on KP20k. $N = 0$ means do not apply segmentation.

N	0	25	50	100	250	500
$F_1@M$	0.364	0.366	0.368	0.362	0.363	0.362
$F_1@5$	0.300	0.303	0.304	0.297	0.298	0.297
#Avg. Len	3.96	4.05	4.00	3.96	3.92	3.92

Table 7. Statistics of present keyphrases predictions with different attention mask matrices. α, β, and γ indicates hyperparameters defined in Eq. (8). "baseline" indicates standard Transformer with attention mask $\mathbf{M}^{full}$.

# Layer	α	β	γ	$F_1@M$	$F_1@5$	$C@M$	$C@5$	#Avg. Len
4	Baseline			0.364	0.300	24,154	23,827	3.96
4	1	1	1	0.372	0.304	24,812	24,042	3.85
			0	0.367	0.302	24,905	23,929	3.95
		0		0.363	0.298	25,051	23,662	4.05
	0			0.370	0.298	24,315	23,632	3.73
6	1	1	1	0.372	0.304	24,562	23,979	3.82
			0	0.364	0.306	25,112	24,208	4.13
		0		0.366	0.296	24,177	23,347	3.90
	0			0.368	0.302	24,468	23,770	3.87

Typically, the mandatory prohibition promotes the encoder to focus more on modeling tokens within the same segment while forcibly prohibiting attending tokens from other segments containing uninformative noises. In contrast to this, a CNN model mainly captures features from the local context due to spatial convolutional mechanism, while a RNN model models the input sequentially. Moreover, previous work [8,15] has shown that in RNN models each token can only perceive approximately 50 to 100 tokens before it. In summary, this observation also provides empirical evidence to the information sparsity hypothesis.

As shown in Table 7, sparsing the attention mask matrix can boost the performance compared to a standard Transformer with a $\mathbf{M}^{full}$ attention mask. Compared to $\mathbf{M}^{full}$, attention with mask matrix $\bar{\mathbf{M}}$ can not only predict more correct keyphrases but also predict fewer keyphrases on average.

According to our ablation experiments, we found that the evaluation score drops the least when $\alpha = 0$, leading tokens containing the least important information among three types of tokens. Meanwhile, prohibiting the neighborhood tokens will lead to a significant drop, which shows that the most informative content usually comes from neighborhood tokens. This observation provides empirical evidence that enhancing the ability to capture local information can make Transformer performs better on KG tasks.

Table 8. Comparison of present keyphrase prediction results. "SM" indicates sparsing the mask matrix of self-attention, and "RMHA" indicates using relative multi-head attention. We **bold** the best result.

Model	Inspec		Krapivin		SemEval		KP20k	
	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$
catSeq [31]	0.276	0.233	0.344	0.269	0.313	0.262	0.368	0.295
ExHiRD [4]	0.291	0.253	0.347	**0.286**	0.335	0.284	0.374	0.311
ExHiRD with RNN (our implementation)	0.288	0.248	0.344	0.281	0.326	0.274	0.374	0.311
(Ours) ExHiRD with TF	0.278	0.232	0.329	0.272	0.310	0.258	0.364	0.300
+ SM only	0.280	0.235	0.334	0.275	0.319	0.266	0.372	0.304
+ RMHA only	0.289	0.244	0.336	0.277	0.325	0.278	0.372	0.313
+ SM + RMHA	**0.293**	**0.254**	**0.351**	**0.286**	**0.337**	**0.289**	**0.375**	**0.316**

Applying Relative Multi-head Attention. As shown in Table 8, applying relative multi-head attention can improve the performance of Transformer models. Meanwhile, it also shows that our model with a sparse self-attention mask matrix can be further improved by applying relative multi-head attention.

4.4 Observations and Findings

In summary of our experiment results, we have four main findings: (1) Our experiments have confirmed that the vanilla Transformer encoder-decoder model can hardly beat the recurrent-based models. (2) In KG tasks, the most informative content usually neither comes from leading tokens nor comes from tokens with the highest attention score, but comes from neighborhood tokens. (3) In KG tasks, direction information is also important, which is not significantly in the position embedding. We boost our model with relative multi-head attention. (4) Our experiments provide empirical evidence to our *information sparsity hypothesis*. Our hypothesis also encourages the adaptation of Transformer models to achieve SOTA results.

5 Related Work

Keyphrases are short phrases that contain the core information summarized from the given document. Traditional extractive approaches [22,29] aim to select salience phrase presents in a document. The existing methods generally adopt two steps. First, the identify keyphrase candidates are extracted by heuristic methods [17]. Afterward, the candidates are ranked by either unsupervised methods [26] or supervised learning techniques [11,23]. In order to generate both present and absent keyphrase for a document, [21] introduced CopyRNN, which consists of an attentional encoder-decoder model [1] and a copy mechanism [9]. CopyRNN and its variants use the beam search to generate a fixed-size keyphrase. [31] introduced a new Seq2Seq model that predicts multiple keyphrases for the given document, which enable a generative model to generate variable numbers of keyphrases. Lately, [3,20] proposed reinforcement learning

based fine-tuning method for generating both sufficient and accurate keyphrases. Furthermore, [4] proposes an exclusive hierarchical decoding framework to avoid repeated keyphrases and enhance the diversity of the generated keyphrases.

6 Conclusion

In this paper, we confirm the poorly performance Transformer models have in KG tasks and seek to explore the reason. We state the Information Sparsity Hypothesis and conduct experiments to confirm our hypothesis. Based on our hypothesis, we adapt the Transformer model by sparsing the attention mask matrix and introducing relative multi-head attention, which achieves SOTA results on KG benchmarks. Ablation study also proves that in KG task the most informative content usually comes from neighborhood tokens. Our detailed observations also provide more hints for the follow-up researchers to understand the Transformer architecture and design more powerful models.

Acknowledgments. This work was supported by the National Key Research and Development Program of China (No. 2020AAA0106700) and National Natural Science Foundation of China (No. 62022027).

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)
2. Berend, G.: Opinion expression mining by exploiting keyphrase extraction. In: IJCNLP, pp. 1162–1170, November 2011
3. Chan, H.P., Chen, W., Wang, L., King, I.: Neural keyphrase generation via reinforcement learning with adaptive rewards. In: ACL, pp. 2163–2174, July 2019
4. Chen, W., Chan, H.P., Li, P., King, I.: Exclusive hierarchical decoding for deep keyphrase generation. In: ACL, pp. 1095–1105, July 2020
5. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP, pp. 1724–1734, October 2014
6. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., Salakhutdinov, R.: Transformer-XL: attentive language models beyond a fixed-length context. In: ACL (2019)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL, June 2019
8. Domhan, T.: How much attention do you need? A granular analysis of neural machine translation architectures. In: ACL, pp. 1799–1808, July 2018
9. Gu, J., Lu, Z., Li, H., Li, V.O.: Incorporating copying mechanism in sequence-to-sequence learning. In: ACL, pp. 1631–1640, August 2016
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997)
11. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: EMNLP, pp. 216–223 (2003)
12. Hulth, A., Megyesi, B.B.: A study on automatically extracted keywords in text categorization. In: ACL, pp. 537–544. ACL-44 (2006)

13. Jones, S., Staveley, M.S.: Phrasier: a system for interactive document retrieval using keyphrases. In: SIGIR, pp. 160–167 (1999)
14. Kim, S.N., Medelyan, O., Kan, M.Y., Baldwin, T.: SemEval-2010 task 5: automatic keyphrase extraction from scientific articles. In: SemEval 2010, pp. 21–26 (2010)
15. Koehn, P., Knowles, R.: Six challenges for neural machine translation. In: Proceedings of the First Workshop on NMT, pp. 28–39, August 2017
16. Krapivin, M., Autaeu, A., Marchese, M.: Large dataset for keyphrases extraction. Technical report, University of Trento (2009)
17. Le, T.T.N., Nguyen, M.L., Shimazu, A.: Unsupervised keyphrase extraction: introducing new kinds of words to keyphrases. In: Kang, B.H., Bai, Q. (eds.) AI 2016: Advances in Artificial Intelligence, pp. 665–671 (2016)
18. Lin, T., Wang, Y., Liu, X., Qiu, X.: A survey of transformers. arXiv preprint arXiv:2106.04554 (2021)
19. Liu, Y., Lapata, M.: Text summarization with pretrained encoders. arXiv preprint arXiv:1908.08345 (2019)
20. Luo, Y., Xu, Y., Ye, J., Qiu, X., Zhang, Q.: Keyphrase generation with fine-grained evaluation-guided reinforcement learning. arXiv preprint arXiv:2104.08799 (2021)
21. Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., Chi, Y.: Deep keyphrase generation. In: ACL, pp. 582–592, July 2017
22. Mihalcea, R., Tarau, P.: TextRank: bringing order into text. In: EMNLP (2004)
23. Nguyen, T.D., Kan, M.Y.: Keyphrase extraction in scientific publications. In: International Conference on Asian Digital Libraries, pp. 317–326 (2007)
24. Shaw, P., Uszkoreit, J., Vaswani, A.: Self-attention with relative position representations. In: Walker, M.A., Ji, H., Stent, A. (eds.) NAACL-HLT (2018)
25. Vaswani, A., et al.: Attention is all you need. In: NeurIPS (2017)
26. Wan, X., Xiao, J.: Single document keyphrase extraction using neighborhood knowledge. In: AAAI, pp. 855–860 (2008)
27. Wang, L., Cardie, C.: Domain-independent abstract generation for focused meeting summarization. In: ACL, pp. 1395–1405, August 2013
28. Wilson, T., Wiebe, J., Hoffmann, P.: Recognizing contextual polarity in phrase-level sentiment analysis. In: EMNLP, pp. 347–354, October 2005
29. Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: KEA: practical automatic keyphrase extraction. In: Proceedings of the Fourth ACM Conference on Digital Libraries, DL 1999, New York, NY, USA, pp. 254–255 (1999)
30. Yan, H., Deng, B., Li, X., Qiu, X.: TENER: adapting transformer encoder for name entity recognition. arXiv preprint arXiv:1911.04474 (2019)
31. Yuan, X., et al.: One size does not fit all: generating and evaluating variable number of keyphrases. In: ACL, pp. 7961–7975, July 2020
32. Zaheer, M., et al.: Big bird: transformers for longer sequences. arXiv preprint arXiv:2007.14062 (2020)