PromptRank: Unsupervised Keyphrase Extraction Using Prompt

Aobo Kong[1] **Shiwan Zhao**[2] **Hao Chen**[3] **Qicheng Li**[1*] **Yong Qin**[1]
Ruiqi Sun[3] **Xiaoyan Bai**[3]
[1]TMCC, CS, Nankai University [2]Independent Researcher
[3]Enterprise & Cloud Research Lab, Lenovo Research
[1]kongaobo9@163.com [2]zhaosw@gmail.com
[1]{liqicheng, qinyong}@nankai.edu.cn
[3]{chenhao31, sunrq2, baixy8}@lenovo.com

Abstract

The keyphrase extraction task refers to the automatic selection of phrases from a given document to summarize its core content. State-of-the-art (SOTA) performance has recently been achieved by embedding-based algorithms, which rank candidates according to how similar their embeddings are to document embeddings. However, such solutions either struggle with the document and candidate length discrepancies or fail to fully utilize the pre-trained language model (PLM) without further fine-tuning. To this end, in this paper, we propose a simple yet effective unsupervised approach, PromptRank, based on the PLM with an encoder-decoder architecture. Specifically, PromptRank feeds the document into the encoder and calculates the probability of generating the candidate with a designed prompt by the decoder. We extensively evaluate the proposed PromptRank on six widely used benchmarks. PromptRank outperforms the SOTA approach MDERank, improving the $F1$ score relatively by 34.18%, 24.87%, and 17.57% for 5, 10, and 15 returned results, respectively. This demonstrates the great potential of using prompt for unsupervised keyphrase extraction. We release our code at this url.

1 Introduction

Keyphrase extraction aims to automatically select phrases from a given document that serve as a succinct summary of the main topics, assisting readers in quickly comprehending the key information, and facilitating numerous downstream tasks like information retrieval, text mining, summarization, etc. Existing keyphrase extraction work can be divided into two categories: supervised and unsupervised approaches. With the development of deep learning, supervised keyphrase extraction methods have achieved great success by using advanced architectures, such as LSTM (Alzaidy et al., 2019; Sahrawat et al., 2020) and Transformer (Santosh et al., 2020; Nikzad-Khasmakhi et al., 2021; Martinc et al., 2022). However, supervised methods require large-scale labeled training data and may generalize poorly to new domains. Therefore, unsupervised keyphrase extraction methods, mainly including statistics-based (Florescu and Caragea, 2017a; Campos et al., 2020b), graph-based (Bougouin et al., 2013; Boudin, 2018), and embedding-based methods (Bennani-Smires et al., 2018; Zhang et al., 2022), are more popular in industry scenarios.

Recent advancements in embedding-based approaches have led to SOTA performances that can be further divided into two groups. The first group of methods, such as EmbedRank (Bennani-Smires et al., 2018) and SIFRank (Sun et al., 2020), embed the document and keyphrase candidates into a latent space, calculate the similarity between the embeddings of the document and candidates, then select the top-K most similar keyphrases. Due to the discrepancy in length between the document and its candidates, these approaches perform less than optimal and are even worse for long documents. To mitigate such an issue, the second kind of approach is proposed. By leveraging a pre-trained language model (PLM), MDERank (Zhang et al., 2022) replaces the candidate's embedding with that of the masked document, in which the candidate is masked from the original document. With the similar length of the masked document and the original document, their distance is measured, and the greater the distance, the more significant the masked candidate as a keyphrase. Though MDERank solves the problem of length discrepancy, it faces another challenge: PLMs are not specifically optimized for measuring such distances so contrastive fine-tuning is required to further improve the performance. This places an additional burden on training and deploying keyphrase extraction systems. Furthermore, it hinders the rapid adoption of large language models when more powerful PLMs

*Qicheng Li is the corresponding author.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 9788–9801

emerge.

Inspired by the work CLIP (Radford et al., 2021), in this paper, we propose to expand the candidate length by putting them into a well-designed template (i.e., prompt). Then to compare the document and the corresponding prompts, we adopt the encoder-decoder architecture to map the input (i.e., the original document) and the output (i.e., the prompt) into a shared latent space. The encoder-decoder architecture has been widely adopted and has achieved great success in many fields by aligning the input and output spaces, including machine translation (Vaswani et al., 2017), image captioning (Xu et al., 2015), etc. Our prompt-based unsupervised keyphrase extraction method, dubbed **PromptRank**, can address the aforementioned problems of existing embedding-based approaches simultaneously: on the one hand, the increased length of the prompt can mitigate the discrepancy between the document and the candidate. On the other hand, we can directly leverage PLMs with an encoder-decoder architecture (e.g., T5 (Raffel et al., 2020)) for measuring the similarity without any fine-tuning. Specifically, after selecting keyphrase candidates, we feed the given document into the encoder and calculate the probability of generating the candidate with a designed prompt by the decoder. The higher the probability, the more important the candidate.

To the best of our knowledge, PromptRank is the first to use prompt for unsupervised keyphrase extraction. It only requires the document itself and no more information is needed. Exhaustive experiments demonstrate the effectiveness of PromptRank on both short and long texts. We believe that our work will encourage more study in this direction.

The main contributions of this paper are summarized as follows:

• We propose PromptRank, a simple yet effective method for unsupervised keyphrase extraction which ranks candidates using a PLM with an encoder-decoder architecture. According to our knowledge, this method is the first to extract keyphrases using prompt without supervision.

• We further investigate the factors that influence the ranking performance, including the candidate position information, the prompt length, and the prompt content.

• PromptRank is extensively evaluated on six widely used benchmarks. The results show that PromptRank outperforms the SOTA approach MDERank by a large margin, demonstrating the great potential of using prompt for unsupervised keyphrase extraction.

2 Related Work

Unsupervised Keyphrase Extraction. Mainstream unsupervised keyphrase extraction methods are divided into three categories (Papagiannopoulou and Tsoumakas, 2020): statistics-based, graph-based, and embedding-based methods. Statistics-based methods (Won et al., 2019; Campos et al., 2020a) rank candidates by comprehensively considering their statistical characteristics such as frequency, position, capitalization, and other features that capture the context information. The graph-based method is first proposed by TextRank (Mihalcea and Tarau, 2004), which takes candidates as vertices, constructs edges according to the co-occurrence of candidates, and determines the weight of vertices through PageRank. Subsequent works, such as SingleRank (Wan and Xiao, 2008), TopicRank (Bougouin et al., 2013), PositionRank (Florescu and Caragea, 2017b), and MultipartiteRank (Boudin, 2018), are improvements on TextRank. Recently, embedding-based methods have achieved SOTA performance. To name a few, EmbedRank (Bennani-Smires et al., 2018) ranks candidates by the similarity of embeddings between the document and the candidate. SIFRank (Sun et al., 2020) follows the idea of EmbedRank and combines sentence embedding model SIF (Arora et al., 2017) and pre-trained language model ELMo (Peters et al., 2018) to get better embedding representations. However, these algorithms perform poorly on long texts due to the length mismatch between the document and the candidate. MDERank (Zhang et al., 2022) solves the problem by replacing the embedding of the candidate with that of the masked document but fails to fully utilize the PLMs without fine-tuning. To address such problems, in this paper, we propose PromptRank which uses prompt learning for unsupervised keyphrase extraction. In addition to statistics-based, graph-based, and embedding-based methods, AttentionRank (Ding and Luo, 2021) calculates self-attention and cross-attention using a pre-trained language model to determine the importance and semantic relevance of a candidate within the document.

Prompt Learning. In the field of NLP, prompt learning is considered a new paradigm to replace

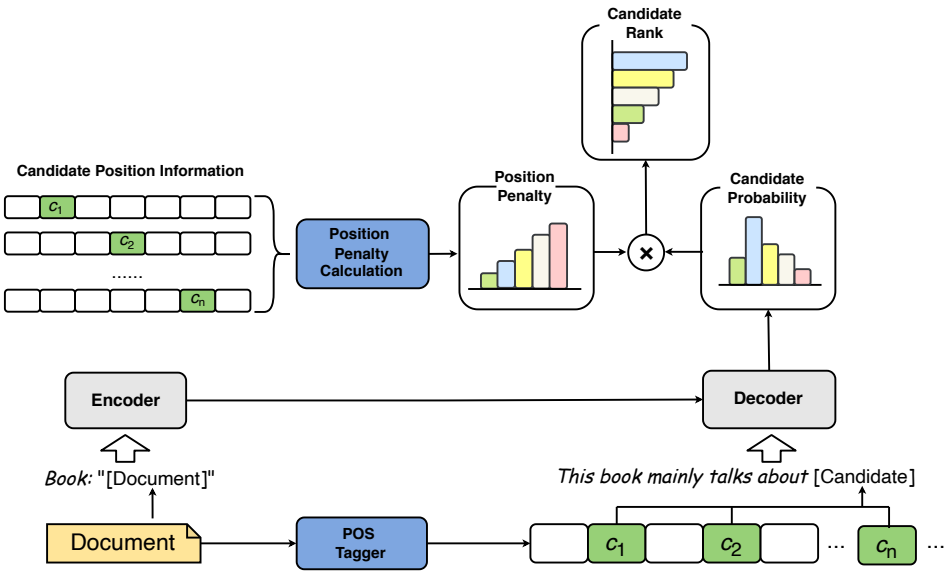


Figure 1: The core architecture of the proposed PromptRank.

fine-tuning pre-trained language models on downstream tasks (Liu et al., 2021). Compared with fine-tuning, prompt, the form of natural language, is more consistent with the pre-training task of models. Prompt-based learning has been widely used in many NLP tasks such as text classification (Gao et al., 2021; Schick and Schütze, 2021), relation extraction (Chen et al., 2022), named entity recognition (Cui et al., 2021), text generation (Li and Liang, 2021), and so on. In this paper, we are the first to use prompt learning for unsupervised keyphrase extraction, leveraging the capability of PLMs with an encoder-decoder architecture, like BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). Our work is also inspired by CLIP (Radford et al., 2021), using the prompt to increase the length of candidates and alleviate the length mismatch.

3 PromptRank

In this section, we introduce the proposed PromptRank in detail. The core architecture of our method is shown in Figure 1. PromptRank consists of four main steps as follows: (1) Given a document d, generate a candidate set $C = \{c_1, c_2, \ldots, c_n\}$ based on part-of-speech sequences. (2) After feeding the document into the encoder, for each candidate $c \in C$, calculate the probability of generating the candidate with a designed prompt by the decoder, denoted as p_c. (3) Use position information to calculate the position penalty of c, denoted as r_c. (4) Calculate the final score s_c based on the probability and the position penalty, and then rank candidates by their s_c in descending order.

3.1 Candidates Generation

We follow the common practice (Bennani-Smires et al., 2018; Sun et al., 2020; Zhang et al., 2022) to extract noun phrases as keyphrase candidates using the regular expression <NN. *|JJ> * <NN.*> after tokenization and POS tagging.

3.2 Probability Calculation

In order to address the limitations of embedding-based methods as mentioned in Section 1, we employ an encoder-decoder architecture to transform the original document and candidate-filled templates into a shared latent space. The similarity between the document and template is determined by the probability of the decoder generating the filled template. The higher the probability, the more closely the filled template aligns with the document, and the more significant the candidate is deemed to be. To simplify the computation, we choose to place the candidate at the end of the template, so only the candidate's probability needs to be calculated to determine its rank.

A sample prompt is shown in Figure 1. In Section 4.4, we investigate how the length and content of the prompt affect the performance. Specifically, we fill the encoder template with the original document and fill the decoder template with one candidate at a time. Then we obtain the sequence probability $p(y_i \mid y_{<i})$ of the decoder template with the candidate based on PLM. The length-normalized

Dataset	Domain	N_{doc}	L_{doc}	S_{can}	S_{gk}	Gold Keyphrase Distribution				
						1	2	3	4	≥5
Inspec	Science	500	122	15841	4912	13.5	52.7	24.9	6.7	2.2
SemEval2017	Science	493	170	21264	8387	25.7	34.4	17.5	8.8	13.6
SemEval2010	Science	243	190	4355	1506	20.5	53.6	18.9	4.9	2.1
DUC2001	News	308	725	35926	2479	17.3	61.3	17.8	2.5	1.1
NUS	Science	211	7702	25494	2453	26.9	50.6	15.7	4.6	2.2
Krapivin	Science	460	8545	55875	2641	17.8	62.2	16.4	2.9	0.7

Table 1: Statistics of six datasets. N_{doc} denotes the number of documents in each dataset. L_{doc} denotes the average length of documents. S_{can} and S_{gk} denote the total number of candidates and gold keyphrases in each dataset, respectively. Gold Keyphrase Distribution denotes the percentage of keyphrase with different lengths in each dataset.

log-likelihood has been widely used due to its superior performance (Mao et al., 2019; Brown et al., 2020; Oluwatobi and Mueller, 2020). Hence we calculate the probability for one candidate as follows:

$$p_c = \frac{1}{(l_c)^\alpha} \sum_{i=j}^{j+l_c-1} \log p(y_i \mid y_{<i}), \quad (1)$$

where j is the start index of the candidate c, l_c is the length of the candidate c, and α is a hyperparameter used to regulate the propensity of PromptRank towards candidate length. We use p_c whose value is negative to evaluate the importance of candidates in descending order.

3.3 Position Penalty Calculation

When writing an article, it is common practice to begin with the main points of the article. Research has demonstrated that the position of candidates within a document can serve as an effective statistical feature for keyphrase extraction (Florescu and Caragea, 2017b; Bennani-Smires et al., 2018; Sun et al., 2020).

In this paper, we use a position penalty to modulate the log probability of the candidate (as shown in Equation 1) by multiplication. The log probabilities are negative, so a larger value of the position penalty is assigned to unimportant positions. This results in a lower overall score for candidates in unimportant positions, reducing their likelihood of being selected as keyphrases. Specifically, for a candidate c, PromptRank calculates its position penalty as follows:

$$r_c = \frac{pos}{len} + \beta, \quad (2)$$

where pos is the position of the first occurrence of c, len is the length of the document, and β is a parameter with a positive value to adjust the influence of position information. The larger the value of β, the smaller the role of position information in the calculation of the position penalty. That is, when β is large, the difference in contribution to the position penalty r_c between two positions will decrease. Therefore, we use different β values to control the sensitivity of the candidate position.

We also observe that the effectiveness of the position information correlates with the document length. The longer the article, the more effective the position information (discussed in Section 4.4). Therefore, we assign smaller value to β for longer documents. Empirically, we formulate β which depends on the length of the document as follows:

$$\beta = \frac{\gamma}{len^3}, \quad (3)$$

where γ is a hyperparameter that needs to be determined experimentally.

3.4 Candidates Ranking

After obtaining the position penalty r_c, PromptRank calculates the final score as follows:

$$s_c = r_c \times p_c. \quad (4)$$

The position penalty is used to adjust the log probability of the candidate, reducing the likelihood of candidates far from the beginning of the article being selected as keyphrases. We rank candidates by the final score in descending order. Finally, the top-K candidates are chosen as keyphrases.

4 Experiments

4.1 Datasets and Evaluation Metrics

For a comprehensive and accurate evaluation, we evaluate PromptRank on six widely used datasets,

in line with the current SOTA method MDERank (Zhang et al., 2022). These datasets are Inspec (Hulth, 2003), SemEval-2010 (Kim et al., 2010), SemEval-2017 (Augenstein et al., 2017), DUC2001 (Wan and Xiao, 2008), NUS (Nguyen and Kan, 2007), and Krapivin (Krapivin et al., 2009), which are also used in previous works (Bennani-Smires et al., 2018; Sun et al., 2020; Saxena et al., 2020; Ding and Luo, 2021). The statistics of the datasets are summarized in Table 1. Following previous works, we use F_1 on the top 5, 10, and 15 ranked candidates to evaluate the performance of keyphrase extraction. When calculating $F1$, duplicate candidates will be removed, and stemming is applied.

4.2 Baselines and Implementation Details

We choose the same baselines as MDERank. These baselines include graph-based methods such as TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), TopicRank (Bougouin et al., 2013), and MultipartiteRank (Boudin, 2018), statistics-based methods such as YAKE (Campos et al., 2020a), and embedding-based methods such as EmbedRank (Bennani-Smires et al., 2018), SIFRank (Sun et al., 2020), and MDERank(Zhang et al., 2022) itself. We directly use the results of the baselines from MDERank. For a fair comparison, we ensure consistency in both pre-processing and post-processing of PromptRank with MDERank. We also use T5-base (220 million parameters) as our model, which has a similar scale to BERT-base (Devlin et al., 2019) used in MDERank. Additionally, to match the settings of BERT, the maximum length for the inputs of the encoder is set to 512.

PromptRank is an unsupervised algorithm with only two hyperparameters to set: α and γ. PromptRank is designed to have out-of-the-box generalization ability rather than fitting to a single dataset. Hence we use the same hyperparameters to evaluate PromptRank on six datasets. We set α to 0.6 and γ to 1.2×10^8. The effects of these hyperparameters are discussed in Section 4.4.

4.3 Overall Results

Table 2 presents the results of the $F1@5$, $F1@10$, and $F1@15$ scores for PromptRank and the baseline models on the six datasets. The results show that PromptRank achieves the best performance on almost all evaluation metrics across all six datasets, demonstrating the effectiveness of the proposed method. Specifically, PromptRank outperforms the SOTA approach MDERank, achieving an average relative improvement of 34.18%, 24.87%, and 17.57% for $F1@5$, $F1@10$, and $F1@15$, respectively. It is worth noting that while MDERank mainly improves the performance on two super-long datasets (Krapivin, NUS) compared to EmbedRank and SIFRank, our approach, PromptRank, achieves the best performance on almost all datasets. This highlights the generalization ability of our approach, which can work well on different datasets with different length of documents.

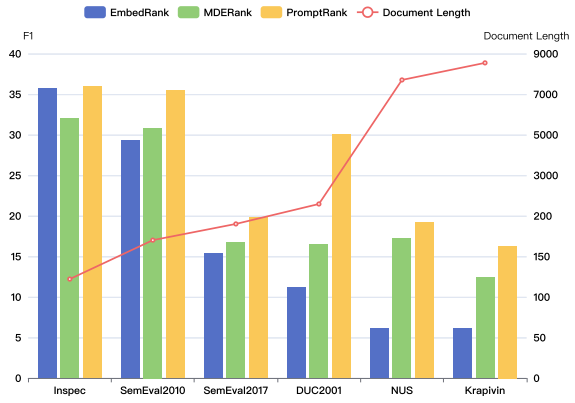


Figure 2: Performance comparison of EmbedRank, MDERank, and PromptRank as the document length increases.

As the document length increases, the length discrepancy between documents and candidates becomes more severe. To further investigate the ability of PromptRank to address this issue, we compare its performance with EmbedRank and MDERank on the average of $F1@5$, $F1@10$, $F1@15$ across the six datasets. As the length of the document increases, the number of candidates increases rapidly, and the performance of keyphrase extraction deteriorates. As shown in Figure 2, EmbedRank is particularly affected by the length discrepancy and its performance drops quickly. Both MDERank and PromptRank mitigate this decline. However, the masked document embedding used in MDERank does not work as well as expected. This is due to the fact that BERT is not trained to guarantee that the more important phrases are masked, the more drastically the embedding changes. BERT is just trained to restore the masked token. By leveraging a PLM of the encoder-decoder structure and using prompt, PromptRank not only more effectively solves the performance degradation of EmbedRank on long texts compared to MDERank but also performs better on short texts than both of them.

$F_1@K$	Method	Dataset						AVG
		Inspec	SemEval2017	SemEval2010	DUC2001	NUS	Krapivin	
5	TextRank	21.58	16.43	7.42	11.02	1.80	6.04	10.72
	SingleRank	14.88	18.23	8.69	19.14	2.98	8.12	12.01
	TopicRank	12.20	17.10	9.93	19.97	4.54	8.94	12.11
	MultipartiteRank	13.41	17.39	10.13	21.70	6.17	9.29	13.02
	YAKE	8.02	11.84	6.82	11.99	7.85	8.09	9.10
	EmbedRank(BERT)	28.92	20.03	10.46	8.12	3.75	4.05	12.56
	SIFRank(ELMo)	29.38	22.38	11.16	24.30	3.01	1.62	15.31
	MDERank(BERT)	26.17	22.81	12.95	13.05	15.24	11.78	17.00
	PromptRank(T5)	**31.73**	**27.14**	**17.24**	**27.39**	**17.24**	**16.11**	**22.81**
10	TextRank	27.53	25.83	11.27	17.45	3.02	9.43	15.76
	SingleRank	21.50	27.73	12.94	23.86	4.51	10.53	16.85
	TopicRank	17.24	22.62	12.52	21.73	7.93	9.01	15.18
	MultipartiteRank	18.18	23.73	12.91	24.10	8.57	9.35	16.14
	YAKE	11.47	18.14	11.01	14.18	11.05	9.35	12.53
	EmbedRank(BERT)	38.55	31.01	16.35	11.62	6.34	6.60	18.41
	SIFRank(ELMo)	**39.12**	32.60	16.03	27.60	5.34	2.52	20.54
	MDERank(BERT)	33.81	32.51	17.07	17.31	18.33	12.93	21.99
	PromptRank(T5)	37.88	**37.76**	**20.66**	**31.59**	**20.13**	**16.71**	**27.46**
15	TextRank	27.62	30.50	13.47	18.84	3.53	9.95	17.32
	SingleRank	24.13	31.73	14.4	23.43	4.92	10.42	18.17
	TopicRank	19.33	24.87	12.26	20.97	9.37	8.30	15.85
	MultipartiteRank	20.52	26.87	13.24	23.62	10.82	9.16	17.37
	YAKE	13.65	20.55	12.55	14.28	13.09	9.12	13.87
	EmbedRank(BERT)	39.77	36.72	19.35	13.58	8.11	7.84	20.90
	SIFRank(ELMo)	**39.82**	37.25	18.42	27.96	5.86	3.00	22.05
	MDERank(BERT)	36.17	37.18	20.09	19.13	17.95	12.58	23.85
	PromptRank(T5)	38.17	**41.57**	**21.35**	**31.01**	**20.12**	**16.02**	**28.04**

Table 2: The performance of keyphrase extraction as $F_1@K$, $K \in \{5, 10, 15\}$ on six datasets.

4.4 Ablation Study

Effects of Position Penalty To evaluate the contribution of the position penalty to the overall performance of PromptRank, we conducted experiments in which candidates were ranked solely based on their prompt-based probability. The results are shown in Table 3. PromptRank without the position penalty outperforms MDERank significantly. When the position penalty is included, the performance is further improved, particularly on long-text datasets. This suggests that prompt-based probability is at the core of PromptRank, and position information can provide further benefits.

Effects of Template Length PromptRank addresses the length discrepancy of EmbedRank by filling candidates into the template. To study how long the template can avoid the drawback of EmbedRank, we conduct experiments using templates of different lengths, namely 0, 2, 5, 10, and 20. Each length contains 4 hand-crafted templates (see details in Appendix A.2), except for the group with length 0, and the position information is not used. To exclude the impact of template content, for each template, we calculate the ratio of the performance of each dataset compared to the dataset Inspec (short text) to measure the degradation caused by an increase in text length. As shown in Figure 3, the higher the polyline is, the smaller the degradation is. Templates with lengths of 0 and 2 degenerate severely, facing the same problem as EmbedRank, making it difficult to exploit prompt. Templates with lengths greater than or equal to 5 better solve the length discrepancy, providing guidance for template selection.

Effects of Template Content The content of the template has a direct impact on the performance of keyphrase extraction. Some typical templates and their results are shown in Table 4 (no position in-

$F_1@K$	Method	Dataset						AVG
		Inspec	SemEval2017	SemEval2010	DUC2001	NUS	Krapivin	
5	PromptRank$_{pt}$	31.79	27.07	16.74	23.71	15.81	14.98	21.68
	PromptRank$_{pt+pos}$	31.73	27.14	17.24	27.39	17.24	16.11	22.81
10	PromptRank$_{pt}$	37.84	37.83	20.82	28.38	18.99	16.35	26.70
	PromptRank$_{pt+pos}$	37.88	37.76	20.66	31.59	20.13	16.71	27.46
15	PromptRank$_{pt}$	38.17	41.82	21.15	28.43	19.59	15.47	27.44
	PromptRank$_{pt+pos}$	38.17	41.57	21.35	31.01	20.12	16.02	28.04

Table 3: The ablation study of position penalty. pt represents the use of prompt-based probability. pos represents the use of the position information.

Number	Encoder	Decoder	$F1@K$		
			5	10	15
1	Book:"[D]"	[C]	14.40	14.41	14.99
2	Book:"[D]"	Keywords of this book are [C]	14.74	20.02	21.81
3	Book:"[D]"	This book mainly focuses on [C]	21.40	26.35	27.06
4	Book:"[D]"	This book mainly talks about [C]	**21.69**	**26.70**	**27.44**
5	Passage:"[D]"	This passage mainly talks about [C]	21.27	26.15	27.25

Table 4: The performance of different templates. [D] is filled with the document and [C] is filled with the candidate. F1 here is the average of six datasets.

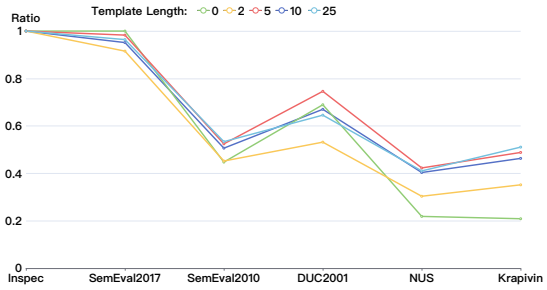


Figure 3: Comparison of the performance decay for different template lengths as the document length increases.

formation used). Template 1 is empty and gets the worst results. Templates 2-5 are of the same length 5 and outperform Template 1. Template 4 achieves the best performance on all metrics. Therefore, we conclude that well-designed prompts are beneficial. Note that all templates are manually designed and we leave the automation of template construction to future work.

Effects of Hyperparameter α The propensity of PromptRank for candidate length is controlled by α. The higher α is, the more PromptRank tends to select long candidates. To explore the effects of different α values, we conduct experiments where the position information is not used. We adjust α from 0.2 to 1, with a step size of 0.1. The optimal values of α on six datasets are shown in Table 5. L_{gk} is the average number of words in gold keyphrases. Intuitively, the smaller L_{gk} of the dataset, the smaller the optimal value of α. Results show that most datasets fit this conjecture. Note that SemEval2017 with the highest L_{gk} is not sensitive to α. The reason is that the distribution of gold keyphrases in the SemEval2017 dataset is relatively more balanced (see table 1). To maintain the generalization ability of PromptRank, it is recommended to select α that performs well on each benchmark rather than pursuing the best average $F1$ across all datasets. Therefore, we recommend setting the value of α to 0.6 for PromptRank.

Effects of Hyperparameter γ The influence of position information is controlled by β in Equation 2. The larger the β, the smaller the impact of the position information on ranking. Previous works (Bennani-Smires et al., 2018; Sun et al., 2020) show that the inclusion of position information can lead to a decrease in performance on short texts while improving performance on long texts. To address this, we dynamically adjust β based on the document length through the hyperparameter γ as shown in Equation 3, aiming to minimize the impact on short texts by a large β

Dataset	L_{gk}	α	β_γ
Inspec	2.31	1	66.08
SemEval2010	2.11	0.5	17.50
SemEval2017	3.00	0.2−1	24.42
DUC2001	2.07	0.4	0.89
NUS	2.03	0.2	0.89
Krapivin	2.07	0.5	0.89

Table 5: Information of hyperparameter setting. $0.2-1$ means the dataset is not sensitive to α. β_γ represents the average values of $beta$ calculated by γ on various datasets and the last three datasets have the same value because of truncation.

Model	$F1@K$		
	5	10	15
T5-small	21.33	25.93	26.52
T5-base	**22.81**	**27.46**	**28.04**
T5-large	22.18	27.11	27.77
BART-base	21.49	25.85	26.63
BART-large	21.86	26.69	27.48

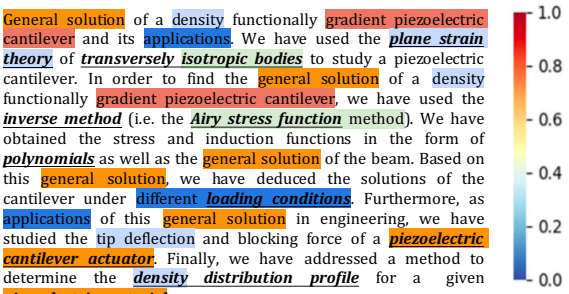
Table 6: The performance using different PLMs. $F1$ here is the average of six datasets.

while maximizing the benefits on long texts by a small β. Through experimentation, we determine the optimal value of γ to be 1.2×10^8. The average values of β calculated via γ on six datasets are shown in Table 5. As shown in Table 3, the performance of PromptRank on short texts remains unchanged while performance on long texts improves significantly.

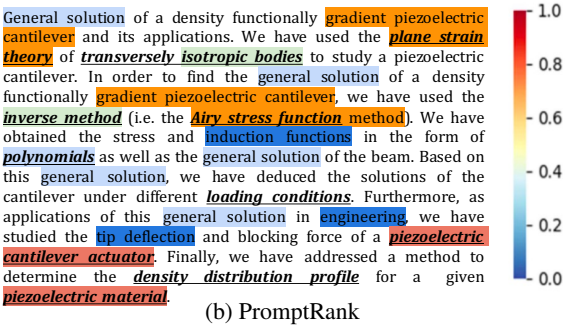
Effects of the PLM PromptRank uses T5-base as the default PLM, but to explore whether the mechanism of PromptRank is limited to a specific PLM, we conduct experiments with models of different sizes and types, such as BART (Lewis et al., 2020). The results, shown in Table 6, indicate that even when the hyperparameters and the prompt are optimized for T5-base, the performance of all models is better than the current SOTA method MDERank. This demonstrates that PromptRank is not limited to a specific PLM and has strong versatility for different PLMs of encoder-decoder structure. Our approach enables rapid adoption of new PLMs when more powerful ones become available.

4.5 Case Study

To demonstrate the effectiveness of PromptRank, we randomly select a document from the Inspec dataset and compare the difference between the scores produced by MDERank and PromptRank in Figure 4. We normalize the original scores and present them in the form of a heat map, where the warmer the color, the higher the score, and the more important the candidate is. Gold keyphrases are underlined in bold italics. The comparison shows that compared to MDERank, PromptRank gives high scores to gold keyphrases more accurately and better distinguishes irrelevant candidates. This illustrates the improved performance of PromptRank over the SOTA method MDERank.



(a) MDERank



(b) PromptRank

Figure 4: Heat maps of candidate keyphrases by MDERank and PromptRank.

5 Conclusion

In this paper, we propose a prompt-based unsupervised keyphrase extraction method, PromptRank, using a PLM of encoder-decoder architecture. The probability of generating the candidate with a designed prompt by the decoder is calculated to rank candidates. Extensive experiments on six widely-used benchmarks demonstrate the effectiveness of our approach, which outperforms strong baselines by a significant margin. We thoroughly examine various factors that influence the performance of PromptRank and gain valuable insights. Additionally, our method does not require any modification to the architecture of PLMs and does not introduce any additional parameters, making it a simple yet powerful approach for keyphrase extraction.

Limitations

The core of PromptRank lies in calculating the probability of generating the candidate with a designed prompt by the decoder, which is used to rank the candidates. Our experiments have shown that the design of the prompt plays a crucial role in determining the performance of the method. While we have manually designed and selected some prompts to achieve state-of-the-art results, the process is time-consuming and may not guarantee an optimal result. To address this limitation, future research could focus on finding ways to automatically search for optimal prompts.

Acknowledgements

The work was supported by National Key R&D Program of China (No.2022ZD0116307), National Natural Science Foundation of China (No. 62271270) and Lenovo Research ECR lab university collaboration program.

References

Rabah Alzaidy, Cornelia Caragea, and C Lee Giles. 2019. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *The world wide web conference*, pages 2551–2557.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 221–229, Brussels, Belgium. Association for Computational Linguistics.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 667–672, New Orleans, Louisiana. Association for Computational Linguistics.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 543–551, Nagoya, Japan. Asian Federation of Natural Language Processing.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020a. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020b. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference 2022*, pages 2778–2788.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Haoran Ding and Xiao Luo. 2021. AttentionRank: Unsupervised keyphrase extraction using self and cross attentions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1928, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Corina Florescu and Cornelia Caragea. 2017a. A new scheme for scoring phrases in unsupervised keyphrase extraction. In *Advances in Information Retrieval*, pages 477–483, Cham. Springer International Publishing.

Corina Florescu and Cornelia Caragea. 2017b. PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115, Vancouver, Canada. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden. Association for Computational Linguistics.

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Huanru Henry Mao, Bodhisattwa Prasad Majumder, Julian McAuley, and Garrison Cottrell. 2019. Improving neural story generation by targeted common sense grounding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5988–5993, Hong Kong, China. Association for Computational Linguistics.

Matej Martinc, Blaž Škrlj, and Senja Pollak. 2022. Tnt-kid: Transformer-based neural tagger for keyword identification. *Natural Language Engineering*, 28(4):409–448.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, pages 317–326, Berlin, Heidelberg. Springer Berlin Heidelberg.

Narjes Nikzad-Khasmakhi, Mohammad-Reza Feizi-Derakhshi, Meysam Asgari-Chenaghlu, Mohammad-Ali Balafar, Ali-Reza Feizi-Derakhshi, Taymaz Rahkar-Farshi, Majid Ramezani, Zoleikha Jahanbakhsh-Nagadeh, Elnaz Zafarani-Moattar, and Mehrdad Ranjbar-Khadivi. 2021. Phraseformer: Multimodal key-phrase extraction using transformer and graph embedding. *arXiv preprint arXiv:2106.04939*.

Olabiyi Oluwatobi and Erik Mueller. 2020. DLGNet: A transformer-based model for dialogue response generation. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 54–62, Online. Association for Computational Linguistics.

Eirini Papagiannopoulou and Grigorios Tsoumakas. 2020. A review of keyphrase extraction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):e1339.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International*

Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. 2020. Keyphrase extraction as sequence labeling using contextualized embeddings. In *European Conference on Information Retrieval*, pages 328–335. Springer.

T.y.s.s Santosh, Debarshi Kumar Sanyal, Plaban Kumar Bhowmick, and Partha Pratim Das. 2020. SaSAKE: Syntax and semantics aware keyphrase extraction from research papers. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5372–5383, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Arnav Saxena, Mudit Mangal, and Goonjan Jain. 2020. KeyGames: A game theoretic approach to automatic keyphrase extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2037–2048, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Yi Sun, Hangping Qiu, Yu Zheng, Zhongwei Wang, and Chaoran Zhang. 2020. Sifrank: A new baseline for unsupervised keyphrase extraction based on pre-trained language model. *IEEE Access*, 8:10896–10906.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860.

Miguel Won, Bruno Martins, and Filipa Raimundo. 2019. Automatic extraction of relevant keyphrases for the study of issue competition. In *Proceedings of the 20th international conference on computational linguistics and intelligent text processing*, pages 7–13.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.

Linhan Zhang, Qian Chen, Wen Wang, Chong Deng, ShiLiang Zhang, Bing Li, Wei Wang, and Xin Cao. 2022. MDERank: A masked document embedding rank approach for unsupervised keyphrase extraction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 396–409, Dublin, Ireland. Association for Computational Linguistics.

A Appendix

A.1 Effects of the Noun Word

We also design experiments to study the impact of the noun word representing the document (no position information used). We consistently use the best-performing template, and only vary the noun word. A total of five different words were tested. As illustrated in Table 7, the choice of noun word does affect the performance of the template, with "Book" achieving the highest results.

A.2 Templates for the Length Study

We use five groups of templates of different lengths to explore the effect of template length. All the templates are shown in Table 8 and $F1$ here is the average of six datasets.

Number	Encoder	Decoder	$F1@K$		
			5	10	15
1	Book:"[D]"	This book mainly talks about [C]	**21.69**	**26.70**	**27.44**
2	Passage:"[D]"	This passage mainly talks about [C]	21.27	26.15	27.25
3	News:"[D]"	This news mainly talks about [C]	20.94	26.09	27.07
4	Text:"[D]"	This text mainly talks about [C]	19.88	25.26	26.43
5	Paper:"[D]"	This paper mainly talks about [C]	21.37	26.43	27.33

Table 7: Templates we design to study the impact of the noun word representing the document.

Length	Encoder	Decoder	$F1@K$		
			5	10	15
0	Book:"[D]"	[C]	14.40	14.41	14.99
2	Book:"[D]"	Book about [C]	15.38	20.88	22.84
2	Book:"[D]"	It is [C]	17.48	23.13	24.87
2	Book:"[D]"	Keywords are [C]	17.48	23.26	24.97
2	Book:"[D]"	Talk about [C]	15.38	20.88	22.84
5	Book:"[D]"	This book are mainly about [C]	21.23	26.28	27.00
5	Book:"[D]"	This book mainly focuses on [C]	21.40	26.35	27.06
5	Book:"[D]"	This book mainly talks about [C]	21.69	**26.70**	**27.44**
5	Book:"[D]"	This book pays attention to [C]	19.33	24.39	25.95
10	Book:"[D]"	All in all, the core of this book is [C]	20.21	25.18	26.27
10	Book:"[D]"	Read this book and tell me that it is about [C]	20.25	25.00	26.46
10	Book:"[D]"	Take a look at the full book, it involves [C]	19.82	25.00	26.31
10	Book:"[D]"	Think carefully, this book has somthing to do with [C]	21.27	26.16	26.93
20	Book:"[D]"	Please read this book carefully from beginning to end and just give your conclusion, this book mainly focuses on [C]	21.11	25.05	25.38
20	Book:"[D]"	The book describes something so interesting, please read it carefully and tell us that this book is about [C]	19.99	24.47	25.36
20	Book:"[D]"	The book is interesting, please read it carefully and summarize its main points with a few keywords like [C]	15.84	20.27	21.23
20	Book:"[D]"	Through careful reading and adequate analysis, we have come to the conclusion that this book mainly talks about [C]	**21.89**	26.44	27.11

Table 8: Templates we design to study the impact of template length.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
We discuss the limitations after the conclusion, and before the references.

☒ A2. Did you discuss any potential risks of your work?
This paper discusses keyphrase extraction, which basically does not bring risks.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
The abstract is at the beginning of the article and the introduction is in Section 1. We summarize the paper's main claims clearly in these two parts.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

The data we use to evaluate the proposed PromptRank is described in Section 4.1. The model PromptRank uses is described in Section 2, 4.2, and 4.4.

☑ B1. Did you cite the creators of artifacts you used?
The data we use to evaluate the proposed PromptRank is cited in Section 4.1. The model PromptRank uses is cited in Section 2, 4.2, and 4.4.

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
The data and the model are so widely used in previous research works. Not discussing the license will not cause ambiguity or bring potential risks. For example, T5 is widely known and is publically available in Transformers (a python library) hence spending space on discussing the license of T5 in the paper is meaningless.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Our use of existing artifacts is consistent with their intended use and there is no potential risk. Spending space on this will make the paper a little strange

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We use the data as previous works did. For example, MDERank, a paper accepted by ACL 2022, does not discuss this. For keyphrase extraction, there is no potential risk.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
The domain of data is shown in Section 4.1.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
The relevant statistics of data are shown in Section 4.1. PromptRank is unsupervised so there are no train/test/dev splits.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

We run computational experiments and discuss relevant information in Section 4.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
We report the number of parameters of T5-base in Section 4.2.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
The setup of hyperparameters and prompts are discussed in Section 4.2 and 4.4.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
We report our results in Section 4.3 and relevant descriptions are clear and accurate. There is no random element in the operation process of our method, so there is no need to discuss whether it is a single run or not.

☒ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
We do use some existing packages like NLTK for stemming or Stanford CoreNLP for pos-tagging. But the use of them does not involve the setting of parameters or something other worth reporting. No relevant description does not affect others to reproduce our work.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.