LongKey: Keyphrase Extraction for Long Documents

Jeovane Honorio Alves
and Radu State
SEDAN - SnT
University of Luxembourg
jeovane.alves,radu.state@uni.lu

Cinthia Obladen
de Almendra Freitas
Graduate Program in Law (PPGD)
Pontifícia Universidade
Católica do Paraná
cinthia.freitas@pucpr.br

Jean Paul Barddal
Graduate Program in Informatics (PPGIa)
Pontifícia Universidade
Católica do Paraná
jean.barddal@ppgia.pucpr.br

Abstract—In an era of information overload, manually annotating the vast and growing corpus of documents and scholarly papers is increasingly impractical. Automated keyphrase extraction addresses this challenge by identifying representative terms within texts. However, most existing methods focus on short documents (up to 512 tokens), leaving a gap in processing long-context documents. In this paper, we introduce LongKey, a novel framework for extracting keyphrases from lengthy documents, which uses an encoder-based language model to capture extended text intricacies. LongKey uses a max-pooling embedder to enhance keyphrase candidate representation. Validated on the comprehensive LDKP datasets and six diverse, unseen datasets, LongKey consistently outperforms existing unsupervised and language model-based keyphrase extraction methods. Our findings demonstrate LongKey's versatility and superior performance, marking an advancement in keyphrase extraction for varied text lengths and domains.

I. INTRODUCTION

Efficient extraction of vital information from textual documents across diverse domains is essential for effective information retrieval, especially given the vast volume of data on the internet and within organizational datasets. In response to this need, Keyphrase Extraction (KPE) aims to identify representative keyphrases that enhance document comprehension, retrieval, and information management [1], [2].

A keyword encapsulates the central theme or a distinct element of a document's subject matter. When multiple words are used, this term is referred to as a *keyphrase*. In practice, the terms *keyword* and *keyphrase* are often used interchangeably. This paper adopts this convention, treating *keyword* and *keyphrase* extraction as synonymous, applicable to terms of any length [3].

Keyphrase extraction techniques are commonly categorized based on their underlying principles [3]. For example, unsupervised methods like TF-IDF [4] calculate term importance based on term frequency within a document and across the corpus. RAKE [5] assesses word relevance through co-occurrence ratios, while TextRank [6] uses a graph-based structure to measure word strength and similarity. KeyBERT [7], unlike unsupervised methods, employs supervised learning with pre-trained BERT embeddings [8] and cosine similarity to determine importance and relevance. PatternRank [9] is similar

to KeyBERT, yet it uses a part-of-speech (POS) module to reduce the number of keyphrase candidates evaluated.

A recent relevant work in keyphrase extraction is Joint-KPE [10], which finetunes a BERT model for keyphrase extraction based on two strategies: global informativeness and keyphrase chunking. Different algorithms served as baselines. ChunkKPE only uses keyphrase chunking as its strategy. Likewise, RankKPE uses only global informativeness as its strategy. TagKPE considers a five-tagging approach to facilitate n-grams extraction. And then SpanKPE, which employs a span self-attention mechanism.

HyperMatch, a new hyperbolic matching model proposed in [11], advances keyphrase extraction beyond Euclidean space, evaluating the relevance of keyphrase candidates using the Poincaré distance. The authors also combine intermediate layers of the RoBERTa [12] model through an adaptive mixing layer to enhance representation. Aimed in long-context documents, GELF [13] is based on graph-enhanced sequence tagging, using the Longformer [14] encoder. The authors constructed a text co-occurrence graph and utilized a graph convolutional network (GCN), focusing on edge prediction, to augment Longformer model embeddings.

Although KPE is a powerful tool, most research has focused on short-context documents, such as abstracts and news articles. While many methods focus on short texts, challenges remain for longer documents. These challenges encompass diverse content structures, increased syntactic complexity, varying contexts within the same document, and limited compatibility with long-context language models. Addressing these intricacies demands developing advanced approaches explicitly tailored for the nuances of handling long-context data [2], [15].

To address these challenges, in this paper, we present LongKey[1], a novel framework that extends keyphrase extraction to long documents through two key contributions. First, LongKey expands token support for encoder models like Longformer, capable of processing up to 96K tokens, ideal for inference on lengthy documents. Second, it introduces a new strategy for keyphrase candidate embedding that captures and

---

[1]Code available at https://github.com/jeohalves/longkey.

consolidates context across the document, enabling a more accurate, context-aware extraction.

The remainder of this paper is organized as follows: Section II details the LongKey methodology, Section III presents the experimental setup, Section IV discusses the results, and Section V concludes the study.

II. PROPOSED APPROACH

Our proposed methodology, dubbed *LongKey*, is outlined in this section. LongKey operates considering three stages: initial word embedding, keyphrase candidate embedding, and candidate scoring, as shown in Figure 1. Each stage is designed to refine the selection and evaluation of keyphrases.

A. Word Embedding

To generate embeddings for long-context documents, our proposal uses the Longformer model [14]. Longformer is an encoder-type language model that uniquely supports extended contexts through two innovative mechanisms: a sliding local windowed attention with a default span of 512 tokens and task-specific global attention mechanism.

By default, each of the model's twelve attention layers produces an output embedding size of 768. Furthermore, Longformer has a positional embedding size of 4,096. We extended it to 8,192 by duplicating the same weights to the next 4,096 elements. For global attention, preliminary experiments have demonstrated optimal results by designating the initial token ([CLS]) as the token for global attention, i.e., the token that attends to every document token and vice-versa.

First, a tokenizer converts the input document to a numeric representation. Our approach uses the Longformer model as the encoder, with tokens defined by the RoBERTa [12] tokenizer. This token representation is then processed by Longformer to generate embeddings, capturing the contextual details of each token within the document.

Even with Longformer, processing of large documents would not be possible with our current computational resources if they are not chunked. Therefore, documents larger than 8K tokens are split in equally sized chunks (with a maximum size of 8192 tokens). Each document is divided into chunks for processing by Longformer, and its embeddings are concatenated to create a unified representation of the entire text's tokens.

Given a document $D = \{w_1, \ldots, w_i, \ldots, w_N\}$ containing N words, we use an encoder-type model to generate the token embeddings E^T:

$$E^T = \text{Encoder}(\{w_1, \ldots, w_i, \ldots, w_N\}). \qquad (1)$$

The resulted operation can be represented as follows:

$$
\begin{aligned}
E^T = \{ & e_{1,1}, e_{1,2}, \ldots, e_{1,M_1}, e_{2,1} \ldots, \\
& e_{i,j}, \ldots, e_{N,M_N}\},
\end{aligned}
\qquad (2)
$$

where $e_{i,j}$ represents the embeddings of the jth token from the ith word in document D. Each embedding $e_{i,j}$ has a size of 768, which is omitted in the explanation for better clarity. If $N > 8192$, D is grouped in chunks which are processed separately and the resulting token embeddings are concatenated together.

B. Keyphrase Embedding

Keyphrase embeddings are context-sensitive, meaning the same keyphrase can yield different embeddings based on its surrounding textual environment. Once these embeddings are crafted, they are combined into unique embeddings for each keyphrase candidate, taking into account the document's overarching thematic and semantic landscape.

Since a specific word may contain more than one token, it's necessary to create a single embedding for this word. Like JointKPE and other similar methods, we used only the first token embeddings to represent the word, since there was no significant difference between this strategy and other simple combinations evaluated, thus reducing computational calculation. These word embeddings are used as the input of our keyphrase embedding module.

Given the token embeddings E^T, the word embeddings are given by preserving only the first token embeddings for each word, given as follows:

$$E^W = \{e_{1,1}, e_{2,1} \ldots, e_{i,1}, \ldots, e_{N,1}\}, \qquad (3)$$

which, for simplicity, we can omit the token index j. Then, we employ a convolutional network to construct embeddings for each potential n-gram keyphrase. For n-grams up to a predetermined maximum length, e.g., $n = 5$, we use n distinct 1-D convolutional layers, each with a kernel size k corresponding to its n-gram size (i.e., $k = n$), ranging from $[1, n]$, and no padding, to generate the keyphrase embeddings from the pre-generated word embeddings. The n-gram representation of the keyphrase occurrence from words w_i to w_{i+k-1} is given by the convolutional module with kernel size k is calculated as follows:

$$h_{i:k} = \text{CNN}^k(\{e_i, \ldots, e_{i+k-1}\}), \qquad (4)$$

where H, the set of keyphrase embeddings, can be represented as:

$$H = \{h_{1:1}, h_{2:1}, \ldots, h_{1:2}, \ldots, h_{i:k}, \ldots, h_{N-k,n}\}. \qquad (5)$$

The convolutional module generates embeddings for each keyphrase occurrence in the text. To capture the relevance of each keyphrase across the document, LongKey uses a keyphrase embedding pooler that combines all occurrences of a keyphrase candidate into a single, comprehensive representation. This approach helps emphasize the most contextually significant keyphrases. A computationally efficient max pooling operation aggregates the diverse embeddings of the keyphrase candidate's occurrences from various text locations into a singular, comprehensive representation. Given KP^n as the set the unique possible keyphrases found in D with maximum size of n words
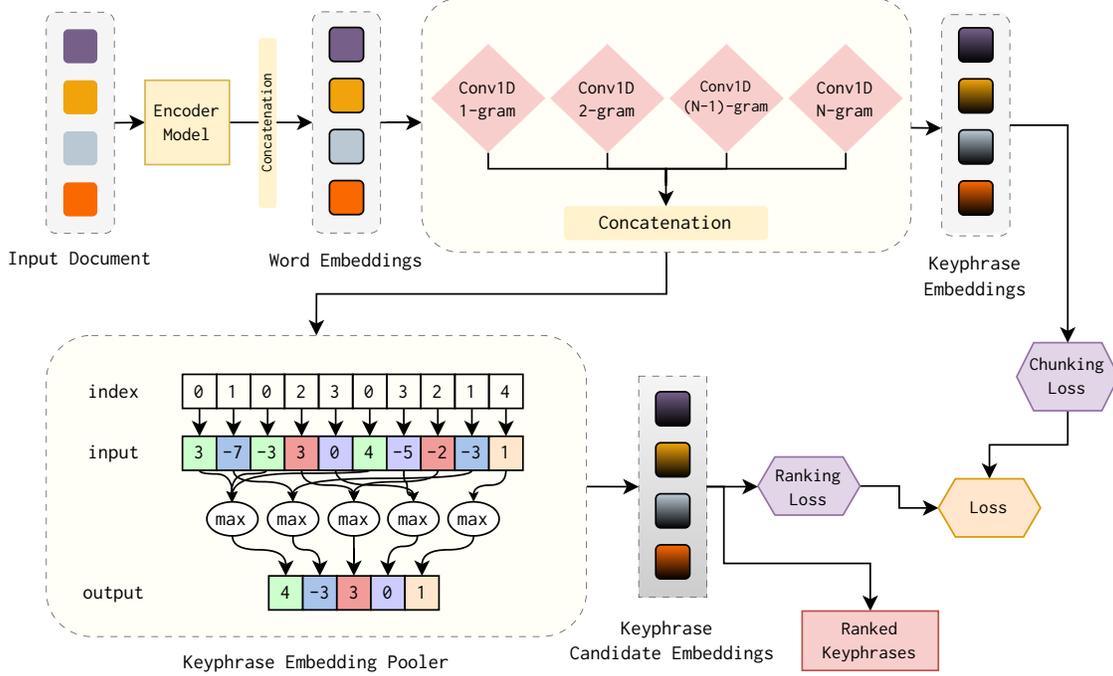
Fig. 1. Overall workflow of the LongKey approach.

$$KP^n = \{kp_1, kp_2, \ldots, kp_i, \ldots, kp_M\}, \qquad (6)$$

where M is the number of unique keyphrases found in D, from unigrams to n-grams, the embeddings of every occurrence of kp_i are defined as follows:

$$H^{KP_l} = \forall h_{i:k} \in H \text{ where } w_{i:k} = KP_l, \qquad (7)$$

thus, for simplicity, H^{KP_l} can be also represented as:

$$H^{KP_l} = \{h_1^l, h_2^l, \ldots, h_i^l, \ldots, h_{S^l}^l\}, \qquad (8)$$

where S^l is the number of occurrences in the document for a specific KP_l. To generate the candidate embeddings C_l for the unique keyphrase l, a max pooling is employed as follows:

$$C^l = \max(\{h_1^l, h_2^l, \ldots, h_i^l, \ldots, h_{S^l}^l\}). \qquad (9)$$

An overall presentation of the candidate embedding calculation is shown in the bottom-left part of Figure 1. For a clear explanation, we show an illustration with the embedding size of 1. This calculation is employed separately for each embedding position. In practice, for each keyphrase candidate, we select its occurrences and get the maximum value. In our example, the keyphrase candidate has three occurrences with values $(3, -3, 4)$ in position $j = 0$. After max pooling, the value for this candidate in the position $j = 0$ is $C_{j=0}^i = 4$. Even though this illustration only has integers, floating-point numbers are employed.

In summary, this cohesive representation encapsulates the essential details from multiple instances, facilitating a robust evaluation of keyphrase relevance for accurate ranking. Consequently, this pooling mechanism strengthens the model's ability to identify the most relevant keyphrases based on context, improving the precision and relevance of the extraction process.

C. Candidate Scoring

In the LongKey approach, candidate embeddings are each assigned a ranking score, with higher scores indicating keyphrases that more accurately represent the document's content. LongKey fine-tunes its performance during training by optimizing ranking and chunking losses, aligning closely with ground-truth keyphrases to ensure relevance. For both losses, ground-truth keyphrases are positive samples. Remaining instances are considered as negative samples.

To generate the scores for both ranking and chunking parts, we employ linear layers for different inputs. For the ranking score, we use the candidate embeddings as the input of a linear layer which converts the embedding to a single value (i.e., ranking score). Given C^l as the embedding of candidate keyphrase l, we calculate the ranking score as follows:

$$s_{\text{rank}}^l = \text{Linear}_{\text{rank}}(C^l) \qquad (10)$$

Unlike JointKPE, which might assign multiple scores to a single keyphrase candidate based on its occurrences, LongKey

assigns a singular score per candidate, facilitated by the efficient proposed keyphrase embedding pooler.

Each candidate's score is then optimized through Margin Ranking loss, enhancing the distinction between positive y^+ and negative y^- samples by elevating the scores of the true keyphrases. This loss is defined as follows:

$$\mathrm{MR}_{\mathrm{loss}}(s_{rank}^+, s_{rank}^-) = \max(0, -s_{rank}^+ + s_{rank}^- + 1) \quad (11)$$

As for the chunking score, we use the keyphrase embeddings as the input of a linear layer. Given H^i as the embedding of a keyphrase i, we calculate the chunking score as follows:

$$s_{chunk}^i = \mathrm{Linear}_{\mathrm{chunk}}(H^i) \quad (12)$$

One thing to note is that LongKey maintains the same objective as JointKPE for keyphrase chunking, utilizing binary classification optimized with Cross-Entropy loss. Given a probability p^+

$$p^+ = \mathrm{Softmax}(s_{\mathrm{chunk}})^+, \quad (13)$$

representing the likelihood of a sample belonging to the positive class, and z, the actual binary class label of the sample (where 1 indicates positive and 0 indicates negative), the BCE loss is calculated using the formula:

$$\mathrm{BCE}_{\mathrm{loss}} = -[z \log(p^+) + (1 - z) \log(1 - p^+)] \quad (14)$$

Both losses are added together and jointly optimized across model training, similar to JointKPE. The formula is given as follows:

$$\mathrm{LongKey}_{\mathrm{loss}} = \mathrm{MR}_{\mathrm{loss}} + \mathrm{BCE}_{\mathrm{loss}} \quad (15)$$

However, distinctively, LongKey diverges from JointKPE in the objectives of its loss functions. Regarding the ranking loss function, LongKey is specifically designed to refine the embeddings of keyphrase candidates, in contrast to JointKPE's focus on optimizing the embeddings of individual keyphrase instances, thereby enhancing the model's overall precision and contextual sensitivity in keyphrase extraction.

III. EXPERIMENTAL SETUP

This section outlines the empirical evaluation of the LongKey method, providing a comprehensive overview of the experimental datasets and the specific configurations underpinning our analysis.

A. Datasets

Robust and large datasets must be employed to train language models and evaluate the capability of an approach in extracting relevant keyphrases from an input document. Many large datasets typically only contain the title and abstract of scientific papers. They are sub-optimal in evaluating long context-based keyphrase extractors since they generally have samples with less than 512 tokens.

Due to the scarcity of datasets containing a high volume of lengthy documents, the *Long Document Keyphrase Identification Dataset* (LDKP) was formulated specifically for extracting keyphrases from full-text papers (which generally surpass 512 tokens) [15]. LDKP has two datasets:

LDKP3K: A variation of the KP20K dataset [16], which contains approximately 100 thousand samples and an average of 6027 words per document.

LDKP10K: A variation of the OAGKx dataset [17], containing more than 1.3M documents, averaging 4384 words per sample.

Other datasets are employed in a zero-shot fashion, i.e., inference only, to assess the capability of different methods trained on both datasets to adapt to different domains and patterns. These datasets are the following:

Krapivin [18]: Features 2,304 full scientific papers from the computer science domain published by ACM.

SemEval2010 [19]: Comprises 244 ACM scientific papers across four distinct sub-domains: distributed systems; information search and retrieval; distributed artificial intelligence – multiagent systems; social and behavioral sciences – economics.

NUS [20]: This dataset contains 211 scientific conference papers with keyphrases annotated by student volunteers, offering a unique perspective on keyphrase relevance.

FAO780 [21]: With 780 documents from the agricultural sector labeled by FAO staff using the AGROVOC thesaurus, this dataset tests the models' performance on domain-specific terminology.

NLM500 [22]: This collection of 500 biomedical papers, annotated with terms from the MeSH thesaurus, assesses the methods' capability in the biomedical domain.

TMC [23]: Including 281 chat logs related to child grooming from the Perverted Justice project, this dataset, with documents and keyphrases based on the formatting from [24], introduces the challenge of informal text and sensitive content.

Although the focus is on long-context documents, it's possible to use the evaluated methods on short documents. To assess the effectiveness of the models trained on the LDKP datasets, we evaluate them on two of the most popular short-context datasets: KP20k and OpenKP:

KP20k [16]: Highly correlated with the LDKP3K dataset, the KP20k is a dataset containing more than 500 thousand abstracts of scientific papers (20 thousand of abstracts for the validation and test subsets each).

OpenKP [25]: The OpenKeyPhrase (OpenKP) is a popular short-context dataset containing more than 140 thousand of real-world web documents, where their keyphrases were human-annotated.

B. Experimental Settings

Our experiments utilized two NVIDIA RTX 3090 GPUs, with 24GB VRAM each. The training regimen was guided by the AdamW optimizer, combined with a cosine annealing learning rate scheduler, with a learning rate value of 5×10^{-5},

and warm-up for the initial 10% training iterations. To circumvent VRAM constraints, we employed gradient accumulation, achieving an effective batch size of 16 in the training phase. To maintain clarity and consistency in our reporting, we use the terms "iterations" and "gradient updates" interchangeably.

We set a maximum token limit of 8,192 during training to accommodate the length of the documents within our available computational resources. The positional embedding was expanded to 8,192, duplicating the original size used by Longformer, which enhances support for longer chunks in inference mode (tested up to 96K in total). We limit keyphrases to a maximum of five words ($k = [1, 5]$) to maintain computational efficiency and align with standard practices in keyphrase extraction. In the evaluation, longer ground-truth keyphrases are considered as false negatives. Moreover, models were trained on LDKP3K for 25 thousand iterations. Since LDKP10K had a substantially higher number of samples, we trained it for 78,125 iterations (i.e., almost an entire epoch). We also evaluated some methods with the BERT model, where we also used chunking to extend training to 8,192 tokens.

To maintain consistency in our analysis and ensure fair comparisons, we used the Longformer model for all supervised approaches that are encoder-based and fine-tuned on the LDKP datasets. Moreover, we employed the same global attention mask as used in LongKey.

Model performance was quantitatively assessed using the F1-score, the harmonic mean between precision and recall, for the most significant K keyphrase candidates (F1@K), with K's value determined based on the overall average of keyphrases per document in each dataset, also following choices of related works, e.g., [26]. Given

$$\hat{Y} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_M] \quad (16)$$

as the predicted keyphrases sorted by their ranking scores in a decreasing order, and Y as the ground-truth keyphrases of a given document (with no specific order), we can calculate the F1-score and its intermediary metrics, i.e., precision and recall; using the top-K predicted keyphrases, given by

$$\hat{Y}_{:k} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_{min(K,M)}]. \quad (17)$$

We calculate the intermediary metrics as follows:

$$\text{Precision@}K = \frac{|\hat{Y}_{:k} \cap Y|}{|\hat{Y}_{:k}|}, \text{Recall@}K = \frac{|\hat{Y}_{:k} \cap Y|}{|Y|}, \quad (18)$$

then, with these two metrics, we calculate the F1-score at the top-K keyphrases as follows:

$$\text{F1@}K = 2 \times \frac{\text{Precision@}K \times \text{Recall@}K}{\text{Precision@}K + \text{Recall@}K}. \quad (19)$$

Another relevant metric, proposed in [27], is a variation of the F1@K defined as F1@$\mathcal{O}$. Here, $\mathcal{O}$ is the number of ground-truth keyphrases (i.e., oracle), thus $K = |Y|$, which is dynamically calculated depending on the document.

This metric is independent to the method's output, given the effectiveness of each method only with the needed predicted keyphrases.

We also employed an additional evaluation: F1@Best. Basically, we evaluate which is the K that have the best harmonic mean between recall and precision, i.e., best F1-score. The purpose of this additional evaluation is to verify how far is the optimum K for a specific method in a specific dataset is from the selected Ks. We put a threshold of $K \leq 100$ to not deviate strongly from the default Ks.

Furthermore, we employed the Porter Stemmer, from the NLTK package [28], for all experiments, but no lemmatization was applied. Stemming was applied for both candidate and ground-truth keyphrases. Duplicated ground-truth keyphrases were cleaned, removing the possibility of duplicated keyphrases erroneously improving the F1-score.

IV. RESULTS AND DISCUSSION

In this section, we delve into the performance outcomes on two primary datasets, extending our analysis to encompass zero-shot learning scenarios and domain-shift adaptability. Moreover, we unravel the contribution of the keyphrase embedding pooler, performance estimation, and inference on short-context documents.

A. LDKP Datasets

Table I presents the comparative results on the LDKP3K test subset, encompassing both unsupervised methods and models finetuned on the LDKP3K and LDKP10K training subsets. It's noteworthy that, aside from GELF, a standard benchmark model, all fine-tuned methods are tailored adaptations designed to handle extensive texts, utilizing the BERT (only when trained on LDKP3K) and Longformer architecture for enhanced context processing. Our approach was also evaluated without chunking, i.e., max of 8192 tokens, identified as *LongKey8K*.

Among the evaluated methods, LongKey8K emerged as the best, achieving an F1@5 of 39.55% and F1@$\mathcal{O}$ of 41.84%. Remarkably, even under a domain shift when trained on the broader LDKP10K dataset, which includes a more comprehensive array of topics beyond computer science, LongKey maintained its lead with an F1@5 of 31.94% and F1@$\mathcal{O}$ of 32.57%.

Performance metrics on the LDKP10K test subset are also provided in Table I, where LongKey emerges as the leading method, achieving an F1@5 of 41.81%.

While LongKey trained on the LDKP3K dataset outperformed other models trained on the same dataset, it scored significantly lower when compared to its performance on the LDKP10K dataset, indicative of dataset-specific variations in effectiveness. This discrepancy, especially the reduced efficacy on the LDKP10K subset, could be attributed to the significant skew towards computer science papers within the LDKP3K dataset, as detailed in the LDKP study.

Generally, the evaluated methods had superior F1@$\mathcal{O}$ than F1s at specific Ks, suggesting that, for the LDKP datasets, ground-truth keyphrases were ranked higher in prediction.

TABLE I

RESULTS OBTAINED ON LDKP TEST SUBSETS. VALUES IN %. THE BEST SCORES FOR EACH K ARE IN **BOLD**. BEST SCORES ONLY IN A SPECIFIC SECTION ARE <u>UNDERLINED</u>. * GELF SCORE WAS REPORTED IN ITS PAPER WITHOUT A SPECIFIC K VALUE.

F1@K	LDKP3K					LDKP10K				
	@4	@5	@6	$\mathcal{O}$	@Best	@4	@5	@6	$\mathcal{O}$	@Best
TF-IDF	8.64	9.08	9.40	8.75	9.72@9	7.45	7.88	8.12	7.77	8.41@9
TextRank	6.28	6.90	7.19	6.68	8.01@12	5.11	5.47	5.82	5.48	6.54@14
PatternRank	7.50	8.24	8.56	7.33	8.65@8	5.62	6.13	6.46	6.12	7.23@14
Trained on LDKP3K										
GELF*	-	-	-	27.10	-	-	-	-	-	-
SpanKPE	30.27	30.08	29.43	31.08	30.27@4	19.99	20.37	20.39	21.00	20.39@6
TagKPE	34.50	34.52	33.94	36.58	34.52@5	21.48	21.84	21.92	22.56	21.92@6
ChunkKPE	31.43	31.17	30.55	32.81	31.43@4	20.12	20.45	20.50	21.06	20.50@6
RankKPE	36.83	36.61	35.81	38.38	36.83@4	23.14	23.70	23.84	24.31	23.84@6
JointKPE	<u>37.50</u>	<u>37.23</u>	36.54	<u>39.41</u>	37.50@4	23.67	<u>24.23</u>	<u>24.37</u>	<u>24.98</u>	24.37@6
HyperMatch	36.34	36.37	35.78	38.23	36.37@5	23.20	23.64	23.77	24.20	23.77@6
BERT-SpanKPE	29.80	30.00	29.51	31.08	30.00@5	20.94	21.46	21.50	21.97	21.50@6
BERT-TagKPE	34.13	34.15	33.49	36.09	34.15@5	21.03	21.40	21.40	21.87	21.40@5
BERT-ChunkKPE	31.80	31.77	31.35	33.89	31.80@4	19.19	19.68	19.74	20.36	19.74@6
BERT-RankKPE	36.28	36.43	35.53	38.38	36.43@5	23.32	23.77	23.89	24.35	23.89@6
BERT-JointKPE	37.19	<u>37.28</u>	36.59	<u>39.94</u>	37.28@5	23.66	<u>24.25</u>	24.26	<u>25.08</u>	24.26@6
BERT-HyperMatch	36.17	36.31	35.49	38.27	36.31@5	23.63	24.10	24.16	24.74	24.16@6
LongKey	39.50	39.50	**38.57**	**41.84**	39.50@5	25.17	25.78	25.77	26.45	25.78@5
BERT-LongKey	38.67	38.68	37.98	40.43	38.68@5	<u>25.36</u>	<u>26.00</u>	<u>26.10</u>	<u>26.58</u>	26.10@6
LongKey8K	**39.55**	**39.54**	**38.57**	**41.84**	**39.55@4**	25.15	25.75	25.77	26.50	25.77@6
Trained on LDKP10K										
SpanKPE	25.83	25.81	25.49	26.54	25.83@4	32.17	32.21	31.75	34.90	32.21@5
TagKPE	30.06	30.12	29.58	31.48	30.12@5	41.12	40.68	39.64	46.47	41.12@4
ChunkKPE	23.93	23.70	23.11	24.65	23.93@4	36.22	35.42	34.43	40.55	36.22@4
RankKPE	28.20	28.39	28.08	29.04	28.39@5	37.98	38.23	37.89	42.37	38.23@5
JointKPE	29.79	29.78	29.44	30.61	29.79@4	39.86	39.95	39.45	44.73	39.95@5
HyperMatch	27.98	28.21	28.07	29.11	28.21@5	37.44	37.52	37.25	41.67	37.52@5
LongKey	<u>31.84</u>	<u>31.94</u>	<u>31.69</u>	<u>32.57</u>	31.94@5	**41.57**	**41.81**	**41.00**	**47.26**	**41.81@5**

B. Unseen Datasets

Without any finetuning on their respective data, LongKey and related methods were evaluated across six diverse domains, as shown in Tables II and III. Remarkably, LongKey outperformed other methods in nearly all tested datasets, with the exception of SemEval2010 and TMC where its results were slightly below the top performers (HyperMatch and RankKPE, respectively).

The choice of LDKP training dataset–LDKP3K or LDKP10K–significantly influenced performance across the unseen datasets, with LDKP3K-trained models excelling in every dataset with the exception of the NLM500 dataset. Although LDKP10K had broader areas of study, LDPK3K had overall longer samples, with an average of 6,027 words per document against an average of 4,384 words in the LDKP10K. Further studies are encouraged to assess the influence of study areas and sample size.

Another thing to note is that, for the unseen datasets, there was a balance dispute between BERT and Longformer-based methods as the best one, even for LongKey. Although access the robustness of BERT with a chunking approach, it also show room for improvements regarding long-context encoders.

Figure 2 presents the performance of LongKey and Joint-KPE on the LDKP3K dataset, categorized by document length.

Overall, LongKey achieved consistently high scores across different encoder models, while JointKPE's performance was more variable. Notably, LongKey's Longformer model performed better on longer documents, while the BERT model maintained more balanced results across various lengths. Additionally, LongKey showed particularly strong results for documents between 512 and 1024 tokens, suggesting potential areas for optimization when handling even longer documents.

Overall, LongKey's robustness was evident as it consistently outperformed other models in nearly all benchmarks, showcasing its broad applicability and strength in keyphrase extraction across varied domains.

C. Component Analysis

To assess the keyphrase embedding pooler (KEP) contribution, we undertook a component analysis using the LDKP3K validation subset. This analysis involved evaluating the LongKey approach with different aggregation functions, i.e., average, sum and maximum; but also the improvement obtained compared to the JointKPE approach.

We used the configuration outlined in the experimental settings, with each model configuration undergoing 12,500 iterations. Table IV shows each configuration's average and

| | **Unseen datasets** | | | | | | | | | | | |
	Krapivin				**SemEval2010**				**NUS**			
F1@K	**@4**	**@5**	**@6**	**@$\mathcal{O}$**	**@5**	**@10**	**@15**	**@$\mathcal{O}$**	**@5**	**@10**	**@15**	**@$\mathcal{O}$**
TF-IDF	6.30	7.02	7.45	6.40	<u>6.62</u>	8.80	10.07	9.42	<u>10.44</u>	<u>12.22</u>	<u>12.38</u>	<u>11.98</u>
TextRank	4.87	5.26	5.77	5.23	6.53	<u>8.95</u>	<u>10.11</u>	<u>9.54</u>	7.83	10.63	11.73	9.47
PatternRank	<u>6.72</u>	<u>7.17</u>	<u>7.61</u>	<u>6.81</u>	6.24	7.91	9.08	8.16	8.53	9.89	11.15	10.22
Trained on LDKP3K												
GELF*	-	-	-	-	-	16.70	-	-	-	21.50	-	-
SpanKPE	27.59	27.62	27.22	28.62	20.78	24.81	25.42	25.72	29.68	30.47	28.30	33.04
TagKPE	29.87	29.72	29.32	31.01	21.81	24.72	25.14	25.57	28.78	31.25	29.09	32.12
ChunkKPE	27.90	27.74	27.50	28.89	20.32	23.58	23.73	24.29	27.77	28.66	26.84	30.46
RankKPE	32.00	31.82	31.19	33.32	20.43	24.99	25.22	25.53	29.22	31.64	30.30	33.32
JointKPE	<u>32.55</u>	<u>32.42</u>	<u>32.10</u>	<u>33.73</u>	19.08	25.10	25.73	25.80	28.22	31.12	30.54	33.61
HyperMatch	<u>31.22</u>	<u>31.44</u>	<u>31.27</u>	<u>32.79</u>	<u>22.20</u>	<u>26.64</u>	<u>26.75</u>	<u>26.82</u>	<u>31.27</u>	<u>33.53</u>	<u>32.23</u>	<u>35.14</u>
BERT-SpanKPE	27.18	27.16	26.82	28.15	20.78	25.50	25.63	26.45	<u>29.91</u>	30.96	28.34	31.30
BERT-TagKPE	26.20	26.30	25.85	27.33	19.00	22.41	22.63	22.53	<u>27.51</u>	27.81	26.46	30.43
BERT-ChunkKPE	24.79	24.67	24.38	25.72	18.35	21.93	22.13	22.61	26.32	27.70	26.71	27.70
BERT-RankKPE	31.20	31.43	31.04	32.49	20.38	24.95	25.94	25.94	26.07	30.05	29.59	30.95
BERT-JointKPE	32.06	<u>32.17</u>	31.80	33.45	22.45	26.09	25.68	26.91	26.57	30.34	29.62	31.06
BERT-HyperMatch	<u>32.16</u>	32.14	<u>31.79</u>	<u>33.47</u>	**24.35**	**27.62**	<u>26.85</u>	**27.85**	28.98	<u>31.82</u>	<u>31.08</u>	<u>33.27</u>
LongKey	**34.96**	**34.82**	34.21	**36.31**	22.31	26.36	**27.37**	<u>27.74</u>	30.02	**33.32**	32.51	**34.95**
BERT-LongKey	34.67	34.86	34.30	36.07	19.93	24.06	25.34	25.69	24.46	28.60	29.34	29.43
LongKey8K	34.94	34.85	**34.23**	36.29	<u>22.31</u>	<u>26.36</u>	27.31	27.60	**30.09**	33.19	32.47	**34.95**
Trained on LDKP10K												
SpanKPE	24.63	25.13	24.91	25.52	22.02	25.35	26.17	26.29	26.00	28.19	26.48	29.56
TagKPE	26.22	26.57	26.38	27.43	21.54	<u>25.82</u>	26.02	26.59	25.86	27.16	26.63	29.13
ChunkKPE	21.37	21.50	21.23	22.30	18.57	<u>20.97</u>	20.54	20.80	24.56	26.11	24.08	26.85
RankKPE	25.56	26.05	26.15	26.88	16.47	20.58	22.59	22.06	25.18	26.57	26.34	27.78
JointKPE	26.68	27.04	27.11	27.68	18.23	21.69	23.23	23.02	25.43	26.42	25.76	27.81
HyperMatch	25.23	25.70	26.01	26.65	16.94	21.11	23.37	23.26	24.50	26.08	25.60	27.16
LongKey	<u>29.90</u>	<u>30.52</u>	<u>30.20</u>	<u>31.33</u>	<u>22.26</u>	25.77	<u>26.61</u>	<u>26.79</u>	<u>27.93</u>	29.20	28.06	30.34
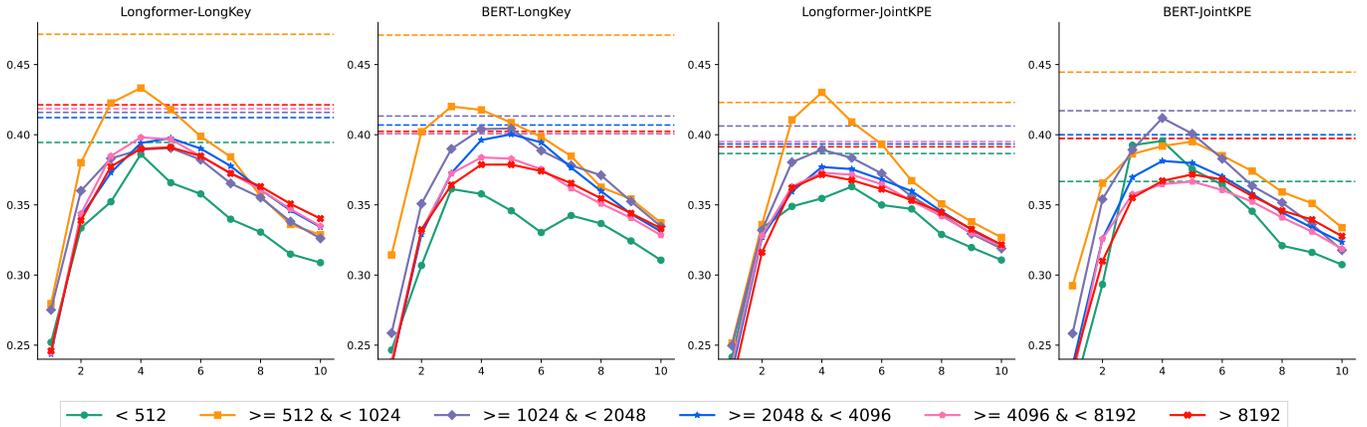
Fig. 2. F1 scores based on the document length of the LongKey and JointKPE methods with different encoders applied to the LDKP3K dataset. F1@K for six range of document length (from less than 512 words to more than 8192), where $K = [1, 10]$. Dashed lines are the F1@$\mathcal{O}$ for the specific interval.

standard deviation results that were computed considering five runs per method.

Overall JointKPE F1@5 score was around 36% with a high std dev. of 0.50%. Using the KEP proposed in LongKey, but with the average reduction, significantly impaired performance, resulting in an F1@5 score of 29.15%, but with a std dev. of 0.23%, lower than JointKPE. Using the summation aggregator improved F1@5 a little (32.76% ± 0.20), but still inferior to JointKPE.

The best F1@5 score was obtained using max pooling,

TABLE III
RESULTS OBTAINED IN UNSEEN DATASETS WITH MODELS TRAINED ON LDKP3K AND LDKP10K TRAINING SUBSETS. VALUES IN %. BEST SCORES, FOR EACH K AND DATASET, ARE IN **BOLD**. BEST SCORES ONLY IN A SPECIFIC SECTION ARE <u>UNDERLINED</u>.

| | Unseen datasets | | | | | | | | | | | |
| | FAO780 | | | | NLM500 | | | | TMC | | | |
F1@K	@4	@5	@6	@𝒪	@5	@10	@15	@𝒪	@40	@50	@60	@𝒪
TF-IDF	7.21	7.97	8.31	8.37	<u>4.66</u>	<u>5.69</u>	<u>5.90</u>	<u>5.61</u>	1.92	2.14	2.41	2.09
TextRank	<u>9.62</u>	<u>10.23</u>	<u>10.65</u>	<u>10.95</u>	4.30	5.35	5.82	5.32	4.83	5.70	6.30	5.35
PatternRank	1.39	1.62	1.84	1.86	2.00	3.21	3.55	2.71	<u>6.91</u>	<u>7.27</u>	<u>7.45</u>	<u>6.93</u>
Trained on LDKP3K												
SpanKPE	15.42	16.23	16.34	16.83	11.15	12.29	11.94	12.34	12.10	13.04	13.63	12.58
TagKPE	18.85	19.31	19.29	20.42	13.57	13.85	13.01	14.34	14.85	15.63	16.24	15.15
ChunkKPE	16.53	17.13	17.41	17.89	11.85	12.52	12.15	12.77	13.63	14.37	14.99	13.94
RankKPE	19.34	19.87	20.42	20.64	14.08	14.78	14.11	15.12	<u>16.21</u>	<u>17.09</u>	<u>17.62</u>	<u>16.53</u>
JointKPE	19.35	<u>19.88</u>	19.98	20.19	<u>14.16</u>	<u>15.14</u>	<u>14.52</u>	<u>15.37</u>	15.26	16.11	16.65	15.38
HyperMatch	<u>19.50</u>	19.81	<u>20.23</u>	<u>20.76</u>	13.64	14.38	13.91	14.59	15.50	16.02	16.16	15.89
BERT-SpanKPE	16.08	16.45	17.03	17.26	11.97	12.35	12.04	12.81	15.26	16.28	16.64	15.92
BERT-TagKPE	17.22	17.77	17.82	18.10	12.88	13.56	13.38	14.34	13.50	14.48	15.10	13.86
BERT-ChunkKPE	13.96	14.49	14.59	14.10	11.90	12.32	11.82	12.42	13.78	14.57	14.94	14.43
BERT-RankKPE	17.26	18.68	19.42	19.25	13.43	13.98	13.75	14.13	**16.80**	**17.44**	**17.78**	**17.75**
BERT-JointKPE	17.58	18.74	18.99	19.29	<u>14.74</u>	<u>14.64</u>	<u>14.11</u>	<u>15.27</u>	15.86	16.51	17.05	16.71
BERT-HyperMatch	<u>18.77</u>	<u>19.25</u>	<u>19.35</u>	<u>20.16</u>	13.11	14.32	13.70	14.72	15.23	16.31	16.81	16.09
LongKey	20.90	21.70	21.87	22.34	14.24	14.96	14.21	15.41	15.89	16.43	16.75	16.20
BERT-LongKey	**22.20**	**22.93**	**22.67**	**23.18**	<u>14.94</u>	<u>15.80</u>	<u>15.04</u>	<u>16.12</u>	16.69	<u>17.31</u>	<u>17.68</u>	<u>17.13</u>
LongKey8K	20.91	21.77	21.84	22.23	14.25	15.00	14.21	15.35	15.92	16.43	16.75	16.26
Trained on LDKP10K												
SpanKPE	17.40	18.07	18.12	18.45	15.00	16.68	16.49	16.63	11.50	12.04	12.31	11.82
TagKPE	19.89	20.72	20.69	21.47	16.23	17.57	17.04	17.52	<u>12.09</u>	<u>12.98</u>	<u>13.75</u>	<u>12.31</u>
ChunkKPE	13.17	13.18	13.00	14.17	13.27	14.13	13.01	14.56	1.20	1.53	1.83	0.82
RankKPE	18.11	19.01	19.45	19.77	15.96	18.94	18.64	**18.86**	8.53	9.47	10.13	9.01
JointKPE	18.03	19.05	19.54	19.88	16.24	17.92	17.67	17.96	9.47	10.49	10.80	9.67
HyperMatch	17.98	18.74	18.95	19.63	14.96	18.43	18.53	18.00	9.95	10.97	11.65	10.33
LongKey	<u>20.00</u>	<u>21.02</u>	<u>21.20</u>	<u>21.92</u>	**16.49**	**19.19**	**18.78**	**18.86**	10.87	11.51	11.80	10.88

TABLE IV
OVERALL RESULTS OBTAINED IN OUR COMPONENT ANALYSIS. SCORES IN %. THE BEST SCORES FOR EACH K ARE IN **BOLD**.

| Component Analysis | | | |
F1@K	@4	@5	@6
JointKPE	36.03±0.50	36.00±0.50	35.24±0.48
+ avg KEP	28.60±0.23	29.15±0.23	29.21±0.34
+ sum KEP	32.54±0.36	32.76±0.20	32.54±0.16
LongKey	**39.04±0.18**	**38.94±0.07**	**38.13±0.10**

the number of processed documents per second using a single RTX 3090. The overall results can be seen in Table V.

As we can see, LongKey performed slightly inferior to the supervised methods. This was basically caused by the keyphrase embedding pooler. However, this performance loss is minor compared with how much the overall F1 increased with the proposed module. Robust approaches with as little bottleneck as possible are encouraged. Also, though in some cases BERT-based methods had inferior results, they have a little boost in performance in comparison with Longformer-based.

E. Short Documents

In Table VI, we show the results of the evaluated methods in two short-context datasets, KP20k and OpenKP. Three methods were generally competitive: RankKPE, JointKPE, and LongKey. Overall, JointKPE was superior on the KP20k (which was originally developed using it). Since KP20k has a high correlation with LDKP3K, better results are expected in models trained with the latter.

For the OpenKP, models trained on the LDKP10K were generally better, especially RankKPE. Here, SpanKPE also had results similar to those of the other three. Overall, LongKey improvements on long-context datasets (except the

achieving almost 39%, with the lowest std. dev. of 0.07%. These findings underscore the KEP's substantial impact on LongKey's success, which is contingent on the appropriate reduction choice.

We suggest that the max aggregator can especially highlight salient features present in different occurrences of a specific keyphrase around the document, thus contributing to a more effective extraction of representative keyphrases.

D. Performance Evaluation

We also evaluate the performance of each method in inference. We calculate the performance for each dataset based on

TABLE V
Performance evaluation of each method tested on each dataset using a single GPU using documents per second. * denotes CPU-only methods.

Performance Evaluation (docs/sec)				
	LDKP3K	LDKP10K	Krapivin	SE2010
TF-IDF*	33.42	41.90	24.89	26.78
TextRank*	3.70	4.26	2.57	2.39
PatternRank	1.35	1.62	1.12	1.10
SpanKPE	1.20	1.79	1.10	1.26
TagKPE	3.96	5.06	3.02	3.03
ChunkKPE	4.11	5.25	3.16	3.15
RankKPE	4.16	5.26	3.22	3.19
JointKPE	4.13	5.25	3.21	3.17
HyperMatch	4.09	5.19	3.22	3.20
BERT-SpanKPE	0.99	1.62	0.71	0.60
BERT-TagKPE	5.29	6.87	3.79	3.70
BERT-ChunkKPE	5.67	7.38	4.26	4.30
BERT-RankKPE	5.83	7.46	4.26	4.34
BERT-JointKPE	5.57	7.11	4.31	4.17
BERT-HyperMatch	5.59	7.14	4.20	4.15
LongKey	4.02	5.06	3.10	3.07
BERT-LongKey	5.59	7.17	4.15	4.17
LongKey8K	4.11	5.20	3.18	3.16
	NUS	FAO780	NLM500	TMC
TF-IDF*	43.35	39.79	41.11	32.50
TextRank*	4.07	2.52	3.36	2.71
PatternRank	1.49	1.47	1.52	1.33
SpanKPE	1.08	1.20	1.56	1.29
TagKPE	4.59	4.61	4.51	3.13
ChunkKPE	4.82	4.82	4.71	3.20
RankKPE	4.84	4.81	4.73	3.04
JointKPE	4.81	4.79	4.66	3.01
HyperMatch	4.78	4.64	4.62	3.06
BERT-SpanKPE	1.13	1.28	1.40	1.26
BERT-TagKPE	5.95	6.45	6.24	2.86
BERT-ChunkKPE	6.70	6.78	6.38	3.98
BERT-RankKPE	6.59	6.90	6.54	3.67
BERT-JointKPE	6.57	6.89	6.46	3.49
BERT-HyperMatch	6.41	6.64	6.09	3.48
LongKey	4.60	4.65	4.54	2.87
BERT-LongKey	6.42	6.59	6.23	3.44
LongKey8K	4.70	4.73	4.66	2.96

documents. LongKey's robustness stems from its innovative architecture, which is specifically designed for long-form content and rigorously validated on extensive datasets crafted for long-context documents.

To validate its efficacy, we conducted a simple component analysis and further assessments of the LDKP datasets, followed by testing across six diverse and previously unseen long-context datasets and two short-context datasets. The empirical results highlight LongKey's capability in long-context KPE, setting a new benchmark for the field and broadening the horizon for its application across extensive textual domains.

Selecting the appropriate LDKP training dataset was crucial for LongKey's performance on unseen data, highlighting the need for strategic modifications to improve generalization without sacrificing the effectiveness of keyphrase extraction. Slightly inferior results in the short-context datasets also indicate the necessity of improvements for a better generalization.

Furthermore, the restriction on the maximum number of words per keyphrase inherently focuses the method on extracting keyphrases of specific lengths. Further adjustments to accommodate longer keyphrases should be explored, as simply increasing keyphrase length may not improve results without careful evaluation. Although this is a common pattern in KPE methods, future work must carefully consider the impact of different keyphrase lengths on overall performance.

Additionally, the context size limitation to 8K tokens – and similarly sized chunks during inference – may restrict LongKey's ability (through not restricted only to our approach) to fully capture and process extensive document content. However, any plans to expand this limit must carefully balance the increased computational demands with available resources.

In summary, LongKey sets a new benchmark in keyphrase extraction for long documents, combining adaptability with high accuracy across various domains. Its superior embedding strategy contributes to its effectiveness, suggesting significant potential for enhancing document indexing, summarization, and retrieval in diverse real-world contexts.

References

[1] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth, "Recent advances in natural language processing via large pre-trained language models: A survey," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–40, 2023.

[2] M. Song, Y. Feng, and L. Jing, "A survey on recent advances in keyphrase extraction from pre-trained language models," *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 2153–2164, 2023.

[3] S. Siddiqi and A. Sharan, "Keyword and keyphrase extraction techniques: a literature review," *International Journal of Computer Applications*, vol. 109, no. 2, 2015.

[4] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, no. 1. Citeseer, 2003, pp. 29–48.

TMC dataset, which has a quite different domain) are not seen in short-context documents. These improvements should be related to the proposed keyphrase embedding pooler. Still, LongKey may also be more biased toward long-context documents, which were not generally seen in the training datasets. Further experiments should be employed, increasing length and content variability in the training stage, to evaluate the capabilities of the keyphrase embedding pooler.

V. Conclusion

Automatic keyphrase extraction is crucial for summarizing and navigating the vast content within documents. Yet, prevalent methods fail to analyze long-context texts like books and technical reports comprehensively. To bridge this gap, we introduce LongKey, a novel keyphrase extraction framework specifically designed for the intricacies of extensive

F1@K	KP20k					OpenKP				
	@3	@4	@5	$\mathcal{O}$	@Best	@3	@4	@5	$\mathcal{O}$	@Best
TF-IDF	<u>15.43</u>	<u>15.22</u>	13.03	<u>15.28</u>	15.45@4	12.48	15.06	<u>13.78</u>	15.17	15.06@3
TextRank	2.94	3.11	2.87	2.97	3.11@5	5.39	7.54	7.56	6.86	7.70@4
PatternRank	13.30	14.96	<u>14.52</u>	12.38	15.19@7	7.40	9.98	9.90	9.49	10.12@4
Trained on LDKP3K										
SpanKPE	30.65	30.31	29.28	32.31	30.65@3	16.87	19.35	17.84	19.88	19.41@2
TagKPE	35.23	34.74	33.59	37.51	35.23@3	15.93	17.42	16.06	18.21	17.52@2
ChunkKPE	33.66	33.11	31.98	35.88	33.66@3	16.05	18.56	17.01	18.68	18.56@3
RankKPE	34.77	34.45	33.35	36.76	34.77@3	16.82	20.31	18.68	20.30	20.31@3
JointKPE	**36.36**	**35.74**	**34.45**	**38.63**	36.36@3	<u>17.24</u>	<u>21.25</u>	<u>19.71</u>	<u>20.89</u>	21.26@2
HyperMatch	35.08	34.59	33.51	37.06	35.08@3	18.58	18.09	17.41	18.20	18.58@3
LongKey	35.32	35.00	33.76	37.21	35.32@3	16.73	20.44	19.13	20.30	20.44@3
Trained on LDKP10K										
SpanKPE	28.40	28.25	27.60	28.84	28.40@3	**19.07**	22.12	20.27	22.61	22.34@2
TagKPE	28.19	28.21	27.66	29.14	28.21@4	18.18	21.30	19.61	22.02	21.42@2
ChunkKPE	25.03	24.56	23.72	26.10	25.03@3	15.57	16.51	14.80	17.38	17.07@2
RankKPE	28.38	28.33	27.74	28.85	28.38@3	18.79	**22.71**	**21.07**	**22.86**	**22.71@3**
JointKPE	<u>29.20</u>	29.06	28.37	29.84	29.20@3	17.84	22.57	21.05	22.32	22.57@3
HyperMatch	28.02	28.35	27.79	28.14	28.35@4	20.60	20.49	19.89	20.02	20.60@3
LongKey	29.19	<u>29.26</u>	<u>28.65</u>	<u>29.87</u>	<u>29.26@4</u>	17.73	22.31	20.90	22.23	22.31@3

[5] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text mining: applications and theory*, pp. 1–20, 2010.

[6] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004, pp. 404–411.

[7] M. Grootendorst, "Keybert: Minimal keyword extraction with bert." 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4461265

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[9] T. Schopf, S. Klimek, and F. Matthes, "Patternrank: Leveraging pre-trained language models and part of speech for unsupervised keyphrase extraction," in *Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KDIR*, INSTICC. SciTePress, 2022, pp. 243–248.

[10] S. Sun, Z. Liu, C. Xiong, Z. Liu, and J. Bao, "Capturing global informativeness in open domain keyphrase extraction," in *Natural Language Processing and Chinese Computing: 10th CCF International Conference, NLPCC 2021, Qingdao, China, October 13–17, 2021, Proceedings, Part II 10*. Springer, 2021, pp. 275–287.

[11] M. Song, Y. Feng, and L. Jing, "Hyperbolic relevance matching for neural keyphrase extraction," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 5710–5720. [Online]. Available: https://aclanthology.org/2022.naacl-main.419

[12] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[13] R. Martínez-Cruz, D. Mahata, A. J. López-López, and J. Portela, "Enhancing keyphrase extraction from long scientific documents using graph embeddings," *arXiv preprint arXiv:2305.09316*, 2023.

[14] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[15] D. Mahata, N. Agarwal, D. Gautam, A. Kumar, S. Parekh, Y. K. Singla, A. Acharya, and R. R. Shah, "LDKP: A Dataset for Identifying Keyphrases from Long Scientific Documents," *arXiv preprint arXiv:2203.15349*, 2022.

[16] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi, "Deep keyphrase generation," *arXiv preprint arXiv:1704.06879*, 2017.

[17] E. Çano and O. Bojar, "Two huge title and keyword generation corpora of research articles," *arXiv preprint arXiv:2002.04689*, 2020.

[18] M. Krapivin, A. Autaeu, M. Marchese *et al.*, "Large dataset for keyphrases extraction," 2009.

[19] S. N. Kim, O. Medelyan, M.-Y. Kan, and T. Baldwin, "Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2010, pp. 21–26.

[20] T. D. Nguyen and M.-Y. Kan, "Keyphrase extraction in scientific publications," in *International conference on Asian digital libraries*. Springer, 2007, pp. 317–326.

[21] O. Medelyan and I. H. Witten, "Domain-independent automatic keyphrase indexing with small training sets," *Journal of the American Society for Information Science and Technology*, vol. 59, no. 7, pp. 1026–1040, 2008.

[22] A. R. Aronson, O. Bodenreider, H. F. Chang, S. M. Humphrey, J. G. Mork, S. J. Nelson, T. C. Rindflesch, and W. J. Wilbur, "The nlm indexing initiative." in *Proceedings of the AMIA Symposium*. American Medical Informatics Association, 2000, p. 17.

[23] A. Kontostathis, L. Edwards, and A. Leatherman, "Text mining and cybercrime," *Text mining: Applications and theory*, pp. 149–164, 2010.

[24] J. H. Alves, H. A. C. G. Pedroso, R. H. Venetikides, J. E. M. Köster, L. R. Grochocki, C. O. A. Freitas, and J. P. Barddal, "Detecting relevant information in high- volume chat logs: Keyphrase extraction for grooming and drug dealing forensic analysis," in *2023 International Conference on Machine Learning and Applications (ICMLA)*, 2023, pp. 1979–1985.

[25] L. Xiong, C. Hu, C. Xiong, D. Campos, and A. Overwijk, "Open domain web keyphrase extraction beyond language modeling," *arXiv preprint arXiv:1911.02671*, 2019.

[26] A. Kong, S. Zhao, H. Chen, Q. Li, Y. Qin, R. Sun, and X. Bai, "Promptrank: Unsupervised keyphrase extraction using prompt," *arXiv preprint arXiv:2305.04490*, 2023.

[27] X. Yuan, T. Wang, R. Meng, K. Thaker, P. Brusilovsky, D. He, and A. Trischler, "One size does not fit all: Generating and evaluating variable number of keyphrases," *arXiv preprint arXiv:1810.05241*, 2018.

[28] E. Loper and S. Bird, "Nltk: The natural language toolkit," *arXiv preprint cs/0205028*, 2002.