MDERank: A Masked Document Embedding Rank Approach for Unsupervised Keyphrase Extraction

Anonymous ACL submission

Abstract

Keyphrase extraction (KPE) automatically extracts phrases in a document that provide a concise summary of the core content, which benefits downstream information retrieval and NLP tasks. Previous state-of-the-art methods select candidate keyphrases based on the similarity between learned representations of the candidates and the document. They suffer performance degradation on long documents due to discrepancy between sequence lengths which causes mismatch between representations of keyphrase candidates and the document. In this work, we propose a novel unsupervised embedding-based KPE approach, Masked Document Embedding Rank (MDERank), to address this problem by leveraging a mask strategy and ranking candidates by the similarity between embeddings of the source document and the masked document. We further develop a KPE-oriented BERT (KPEBERT) model by proposing a novel self-supervised contrastive learning method, which is more compatible to MDERank than vanilla BERT. Comprehensive evaluations on six KPE benchmarks demonstrate that the proposed MDERank outperforms state-of-the-art unsupervised KPE approach by average 1.80 $F1@15$ improvement. MDERank further benefits from KPEBERT and overall achieves average 3.53 $F1@15$ improvement over SIFRank.

1 Introduction

Keyphrase extraction (KPE) automatically extracts a set of phrases in a document that provide a concise summary of the core content. KPE is highly beneficial for readers to quickly grasp the key information of a document and for numerous downstream tasks such as information retrieval and summarization. Previous KPE works include supervised (Liu et al., 2019; Yang et al., 2019; Beltagy et al., 2020; Clark et al., 2020; Zaheer et al., 2020) and unsupervised approaches. Supervised approaches model KPE as sequence tagging or sequence generation tasks and require large-scale annotated data to perform well. Since KPE annotations are expensive and large-scale KPE annotated data is scarce, unsupervised KPE approaches, such as TextRank (Mihalcea and Tarau, 2004), YAKE (Campos et al., 2018), EmbedRank (Bennani-Smires et al., 2018), are the mainstay in industry deployment.

Among unsupervised KPE approaches, embedding-based approaches including EmbedRank (Bennani-Smires et al., 2018) and SIFRank (Sun et al., 2020) yield the state-of-the-art (SOTA) performance. After selecting keyphrase (KP) candidates from a document using rule-based methods, embedding-based KPE approaches rank the candidates in a descending order based on a scoring function, which computes the similarity between embeddings of candidates and the source document. Then the top-K candidates are chosen as the final KPs. We refer to these approaches as *Phrase-Document-based* (PD) methods.

PD methods have two major drawbacks: (i) As a document is typically significantly longer than candidate KPs and usually contains multiple KPs, it is challenging for PD methods to reliably measure their similarities in the latent semantic space. Hence, PD methods are naturally biased towards longer candidate KPs, as shown by the example in Table 1. (ii) The embedding of candidate KPs in the PD methods is computed without the contextual information, hence further limiting the effectiveness of the subsequent similarity match.

In this paper, we propose a novel unsupervised embedding-based KPE method, denoted by

Document	The paper presents a method for pruning frequent itemsets based on background knowledge represented by a Bayesian network . The interestingness of an itemset is defined as the absolute difference between its support estimated from data and from the Bayesian network. Efficient algorithms are presented for finding interestingness of a collection of frequent itemsets , and . . .
SIFRank (Best PD method)	notation database attributes, research track paper dataset #attrs max, bayesian network bn output, bayesian network computing, interactive network structure improvement process
MDERank (Proposed method)	interestingness, pruning, frequent itemsets, pruning frequent itemsets, interestingness measures

Table 1: An example shows the bias of Phrase-Document (PD) methods towards longer candidate keyphrases at $K = 5$. Keyphrase extracted are shown in a ranked order and those matching the gold labels are marked in red.

Masked Document Embedding Rank (**MDERank**), to address above-mentioned drawbacks of PD methods. The architecture of MDERank is shown in Figure 1. The basic idea of MDERank is that a keyphrase plays an important role in the semantics of a document, and its absence from the document should cause a significant change in the semantics of the document. Therefore, we propose to compare the embeddings of the original document and its variant where the occurrence(s) of some candidate KPs are masked. This leads to a new ranking principle based on the *increasing* order of the resulting similarities, i.e., a *lower* semantic similarity between the original document and its masked variant indicates a higher significance of the candidate.

Our proposed method can be deemed as *Document-Document* method and it addresses the two weaknesses of the *Phrase-Document* methods: (i) Since the sequence lengths of the original document and the masked document are the same, comparing their similarities in the semantic space is more meaningful and reliable. (ii) The embedding of the masked document is computed from sufficient amount of context information and hence can capture the semantics reliably using the SOTA contextualized representation models such as BERT. Inspired by (Lewis et al., 2019; Zhang et al., 2020; Han et al., 2021), where pre-trained language models (PLMs) trained on objectives close to final downstream tasks achieve enhanced representations and improve fine-tune performance, we further propose a novel self-supervised contrastive learning method on top of BERT-based models (dubbed as **KPEBERT**).

The main contributions of this work include:

- We propose a novel embedding-based unsupervised KPE approach (MDERank) that improves the reliability of computing KP candidate embeddings from contextualized representation models and improves robustness to different lengths of KPs and documents.

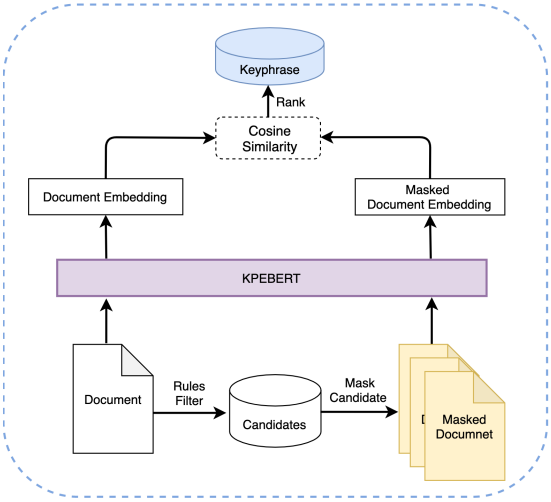

Figure 1: The architecture of the proposed MDERank approach.

- We propose a novel self-supervised contrastive learning method and develop KPEBERT.

- We conduct extensive evaluations of MDERank on six diverse KPE benchmarks and demonstrate the robustness of MDERank to different lengths of documents. MDERank with BERT achieves 17.00, 21.99 and 23.85 for average F1@5, F1@10, F1@15 respectively, as 1.69, 2.18 and 1.80 absolute gains over the SOTA results from SIFRank (Sun et al., 2020), and 4.44, 3.58, and 2.95 absolute gains over EmbedRank with BERT. MDERank with KPEBERT achieves further absolute gains by 1.70, 2.18 and 1.73. Ablation analysis further provides insights on the effects of document lengths, encoder layers, and pooling methods.

2 Related Work

Unsupervised KPE Unsupervised KPE approaches do not require annotated data and there has been much effort in this line of research, as summarized in (Papagiannopoulou and Tsoumakas, 2020). Unsupervised KPE approaches can be categorized into statistics-based, graph-based, and embedding-based methods. The statistics-based YAKE (Campos et al., 2018) method explores both conventional position and frequency features and new statistical features capturing context information. TextRank (Mihalcea and Tarau, 2004) is a representative graph-based method, which converts a document into a graph based on lexical unit co-occurrences and applies PageRank iteratively. Many graph-based methods could be considered as modifications to TextRank by introducing extra features to compute weights for edges of the constructed graph, e.g., SingleRank (Wan and Xiao, 2008), PositionRank (Florescu and Caragea, 2017), ExpandRank (Wan and Xiao, 2008). The graph-based TopicRank (Bougouin et al., 2013) and MultipartiteRank (Boudin, 2018) methods enhance keyphrase diversity by constructing graphs based on clusters of candidate keyphrases. For embedding-based methods, EmbedRank (Bennani-Smires et al., 2018) measures the similarity between phrase and document embeddings for ranking. SIFRank (Sun et al., 2020) improves the static embeddings in EmbedRank by a pre-trained language model ELMo and a sentence embedding model SIF (Arora et al., 2016). As analyzed in Section 1, for embedding-based methods, using contextualized embedding models to compute candidate embeddings could be unreliable due to lack of context, and these methods lack robustness to different lengths of keyphrases and documents. Our proposed MDERank approach could effectively address these drawbacks.

Contextual Embedding Models Early emebdding models include static word embeddings such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and FastText (Bojanowski et al., 2017), and sentence embeddings such as Sent2Vec (Pagliardini et al., 2017) and Doc2Vec (Lau and Baldwin, 2016), which render word or sentence representations that do not depend on their context. In contrast, pre-trained contextual embedding models, such as ELMo (Peters et al., 2018), incorporate rich syntactic and semantic information from context for representation learning and yield more expressive representations. BERT (Devlin et al., 2018) captures better context information through a bidirectional transformer encoder than the Bi-LSTM based ELMo, and has established SOTA in a wide variety of NLP tasks. In one line of research, RoBERTa (Liu et al., 2019), XLNET (Yang et al., 2019) and many other BERT variant PLMs have been proposed to further improve the language representation capability. In another line of research, Longformer (Beltagy et al., 2020), BigBird (Zaheer et al., 2020) and other efficient transformers are proposed to reduce the quadratic complexity of transformer on sequence length in order to model long-range dependencies. In this paper, we mainly use BERT as the default contextual embedding model. We also evaluate the performance of MDERank with these efficient transformers on long documents.

3 MDERank

In this section, we describe the proposed Masked Document Embedding Rank (MDERank) approach. Given a document $d = \{w_1, w_2, \ldots, w_n\}, d \in D$ where D denotes a dataset, and a set of selected candidate KPs $C = \{c_1, \ldots, c_i, \ldots, c_m\}$, where a candidate c_i consists of one or multiple tokens, as $c_i = \{c_i^1, \ldots, c_i^l\}$, and $m \leq n$, KPE aims to select K candidates from C, where $K \leq m$. Following the common practice (Bennani-Smires et al., 2018; Sun et al., 2020), after tokenization and POS tagging, candidates are selected with regular expression `<NN. *|JJ> * <NN.*>`.

To address the mismatch between sequence lengths of a document and a candidate phrase as well as lack of contextual information in PD methods as mentioned in Section 1, we hypothesize that it is more reasonable to conduct the similarity comparison at the *document-document* level rather than at the *phrase-document* level. Based on this hypothesis, for each candidate KP c_i for a document d, given its occurrence positions in d as $[p_1, p_2, \ldots, p_t]$, MDERank replaces all occurrences of $p_{i=1}^t$ by a special placeholder token `MASK` and construct a masked variant of the original document as $d_M^{c_i}$. We define the scoring function $f(c_i)$ for ranking the significance of candidates as the cosine similarity between $E(d)$ and $E(d_M^{c_i})$, where $E(\cdot)$ represents the embedding of a document. Note that a higher $f(c_i)$ value indicates a lower ranking for c_i, which is opposite to the PD methods. We use BERT (Devlin et al., 2018) as the

default embedding model and investigate other contextual embedding models in Section 5.4. BERT is pre-trained through self-supervised tasks of masked language modeling (MLM) and next sentence prediction (NSP), on large-scale unlabeled text of English Wikipedia (2500M words) and Bookscorpus (800 words). A document $d = \{w_1, w_2, \ldots, w_n\}$ is prepended with a special token `[CLS]` and then encoded by BERT to obtain the hidden representations of tokens as $\{H_1, H_2, \ldots, H_n\}$. The document embedding $E(d)$ is computed as follows:

$$E(d) = \text{MaxPool}(H_1, \ldots, H_n)$$

We also investigate average pooling in Section 5.4 and other masking methods in Appendix A.

4 KPEBERT: KPE-oriented Self-supervised Learning

BERT and many other BERT variant PLMs can effectively capture syntactic and semantic information in language representations for downstream NLP tasks, through self-supervised learning objectives such as MLM. However, these self-supervised learning objectives neither explicitly model the significance of KPs nor model ranking between KPs. In this paper, we propose a novel PLM KPEBERT trained with a novel self-supervised learning objective to improve the capabilities of PLMs for ranking KPs. This new task is defined as minimizing the triplet loss between positive and negative examples (See Figure 2). After obtaining a set of pseudo-KPs for documents using another unsupervised KPE method θ, we define documents masking out pseudo-KPs as positive examples while those masking out "non-pseudo-KPs" as negative examples. Following SimCSE (Gao et al., 2021), we encode the original document d (anchor), the positive example d^+, and negative example d^- through a PLM encoder, respectively, and obtain their hidden representations as H_d, H_{d^+}, and H_{d^-}. Finally, we define the triplet loss as

$$\ell_{CL} = \max\left(sim(H_d, H_{d^+}) - sim(H_d, H_{d^-}) + m, 0\right)$$

where $sim(H_x, H_y)$ denotes the similarity between embeddings of the document x and y. We use cosine similarity (same as used for MDERank). m is a margin parameter.

We initialize KPEBERT from BERT-base-uncased[1] and then incorporate the standard MLM pre-training task as in BERT into the overall learning objective to avoid forgetting the previously learned general linguistic knowledge, as follows:

$$\ell = \ell_{CL} + \lambda \cdot \ell_{MLM}$$

where λ is a hyper-parameter balancing the two losses in the multi-task learning setting. KPEBERT differs from SimSCE in two major aspects: (i) KPEBERT uses pseudo labeling and positive/negative example sampling strategies (below), different from standard dropout used by SimCSE to construct pair examples; (ii) KPEBERT uses triplet loss whereas SimCSE uses contrastive loss.

Datasets	N_{KP}	L_{KP}	L_{Doc}
Inspec	9.82	2.31	121.84
SemEval2010	15.07	2.11	189.90
SemEval2017	17.30	3.00	170.38
DUC2001	8.08	2.07	724.63
NUS	11.66	2.07	7702.00
Krapivin	5.74	2.03	8544.57

Table 2: Statistics of the datasets. N_{KP} is the average number of gold keyphrases. L_{KP} is the average length of gold keyphrases. L_{Doc} is the average number of words per document.

Absolute Sampling For a document d, we first select candidate keyphrases C using POS tags with regular expressions as described in Section 3. Then we obtain a set of keyphrases C' extracted by another unsupervised KPE approach θ on d, as pseudo labels. We define "keyphrases" as C' and "non-keyphrases" as $C \setminus C'$. We mask a "keyphrase" from a document with a `MASK` to construct a positive example d^+ for d. We select a "non-keyphrase" and perform the same mask operation to construct a negative example d^-.

Relative Sampling In this approach, after obtaining a set of KP C' extracted by θ, we randomly select a pair of KPs from C' and choose the one ranked higher to construct a positive example and the other one to construct a negative example through the mask operation. On one hand, the decisions of "keyphrases" and "non-keyphrases" are fully based on the ranking predicted by θ, hence relative sampling may increase the impact from θ on the inductive bias of KPEBERT. On the other hand, relative sampling mines more hard negative samples which may improve performance of triplet loss based learning. We study the efficacy of these two sampling approaches on KPEBERT in Section 5.3.

[1] https://huggingface.co/bert-base-uncased

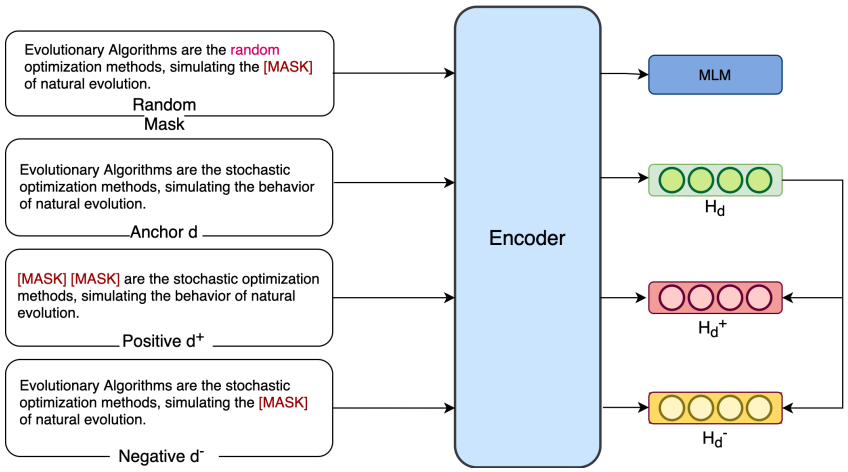

Figure 2: The multi-task pre-training for KPEBERT includes two tasks. One is teaching the encoder to distinguish documents masked with keyphrases and non-keyphrases. The other is further pre-training the encoder with a MLM task. The word in pink is an example to illustrate the random masking in MLM.

$F_1@K$	Method	Dataset						AVG	AvgRank (STD)
		Inspec	SemEval2017	SemEval2010	DUC2001	Krapivin	NUS		
5	TextRank	21.58	16.43	7.42	11.02	6.04	1.80	10.72	9.33 (±1.60)
	SingleRank	14.88	18.23	8.69	19.14	8.12	2.98	12.01	7.67 (±0.94)
	TopicRank	12.20	17.10	9.93	19.97	8.94	4.54	12.11	7.17 (±1.77)
	MultipartiteRank	13.41	17.39	10.13	21.70	9.29	6.17	13.02	6.17 (±1.77)
	YAKE	8.02	11.84	6.82	11.99	8.09	7.85	9.10	9.00 (±2.52)
	EmbedRank(Sent2Vec)+MMR	14.51	20.21	9.63	21.75	8.44	2.13	12.78	6.83 (±2.03)
	SIFRank(ELMo)	**29.38**	**22.38**	11.16	**24.30**	1.62	3.01	15.31	4.50 (±3.77)
	EmbedRank(BERT)	28.92	20.03	10.46	8.12	4.05	3.75	12.56	6.83 (±3.02)
	MDERank(BERT)	26.17	**22.81**	**12.95**	13.05	11.78	**15.24**	17.00	3.33 (±2.49)
	MDERank(KPEBERT$_{ab}$)	28.06	21.63	**12.95**	22.51	**12.91**	14.11	**18.70**	**2.67** (±0.75)
	MDERank(KPEBERT$_{re}$)	27.85	20.37	**13.05**	**23.31**	**12.35**	**14.39**	**18.55**	**2.50** (±1.12)
10	TextRank	27.53	25.83	11.27	17.45	9.43	3.02	15.76	8.00 (±1.63)
	SingleRank	21.50	27.73	12.94	23.86	10.53	4.51	16.85	6.67 (±1.49)
	TopicRank	17.24	22.62	12.52	21.73	9.01	7.93	15.18	8.50 (±1.50)
	MultipartiteRank	18.18	23.73	12.91	24.10	9.35	8.57	16.14	7.17 (±1.67)
	YAKE	11.47	18.14	11.01	14.18	9.35	11.05	12.53	9.17 (±2.54)
	EmbedRank(Sent2Vec)+MMR	21.02	29.59	13.9	25.09	10.47	2.94	17.17	6.67 (±2.29)
	SIFRank(ELMo)	**39.12**	**32.60**	16.03	**27.60**	2.52	5.34	20.54	4.50 (±3.91)
	EmbedRank(BERT)	**38.55**	31.01	16.35	11.62	6.60	6.34	18.41	6.50 (±3.20)
	MDERank(BERT)	33.81	**32.51**	17.07	17.31	12.93	**18.33**	21.99	4.00 (±2.45)
	MDERank(KPEBERT$_{ab}$)	35.80	32.23	**17.95**	**26.97**	**14.36**	17.72	**24.17**	**2.33** (±0.75)
	MDERank(KPEBERT$_{re}$)	34.36	31.21	**18.27**	26.65	**14.31**	**18.46**	**23.88**	**2.50** (±1.26)
15	TextRank	27.62	30.50	13.47	18.84	9.95	3.53	17.32	8.00 (±1.73)
	SingleRank	24.13	31.73	14.4	23.43	10.42	4.92	18.17	6.67 (±1.49)
	TopicRank	19.33	24.87	12.26	20.97	8.30	9.37	15.85	8.83 (±1.77)
	MultipartiteRank	20.52	26.87	13.24	23.62	9.16	10.82	17.37	7.33 (±1.80)
	YAKE	13.65	20.55	12.55	14.28	9.12	13.09	13.87	9.00 (±2.45)
	EmbedRank(Sent2Vec)+MMR	23.79	33.94	14.79	24.68	10.17	3.56	18.49	6.50 (±1.98)
	SIFRank(ELMo)	**39.82**	**37.25**	18.42	**27.96**	3.00	5.86	22.05	4.67 (±3.77)
	EmbedRank(BERT)	**39.77**	36.72	19.35	13.58	7.84	8.11	20.90	6.33 (±3.30)
	MDERank(BERT)	36.17	37.18	20.09	19.13	12.58	**17.95**	23.85	4.00 (±2.00)
	MDERank(KPEBERT$_{ab}$)	37.43	**37.52**	**20.69**	26.28	**13.58**	**17.95**	**25.58**	**2.00** (±1.00)
	MDERank(KPEBERT$_{re}$)	36.40	36.63	**20.35**	**26.42**	**13.31**	**19.41**	**25.42**	**2.67** (±1.37)

Table 3: KPE performance as $F_1@K, K \in \{5, 10, 15\}$ on the six benchmarks. For each K, the first group shows performance of the baselines and the second group shows performance of our proposed MDERank using BERT for embedding and MDERank using KPEBERT for embedding. EmbedRank(BERT) denotes the Phrase-Document based methods for a fair comparison. The best results are both underlined and in bold. The second-best results are in bold. **Ab** and **Re** denote absolute and relative sampling, respectively. **AVG** is the average $F_1@K$ on all six benchmarks. **AvgRank(STD)** is the mean and std of the rank of a method among all methods on all six benchmarks (hence the lower the better).

5 Experiments

5.1 Datasets and Metrics

The pre-training data for KPEBERT are the WikiText language modeling dataset with 100+ million tokens extracted from a set of verified Good and Featured articles on Wikipedia[2]. We use six KPE benchmarks for evaluations. Four of them are scientific publications[3], including Inspec (Hulth, 2003), Krapivin (Krapivin et al., 2009), NUS (Nguyen and Kan, 2007), and SemEval-2010 (Kim et al., 2010), all widely used for evaluations in previous works (Meng et al., 2017; Chen et al., 2019; Sahrawat et al., 2019; Bennani-Smires et al., 2018; Meng et al., 2020). We also evaluate on the DUC2001 dataset (news articles) (Wan and Xiao, 2008) and SemEval2017 dataset (science journals) (Augenstein et al., 2017)[4]. Table 2 shows data statistics. Following previous works, predicted KPs are deduplicated and the KPE performance is evaluated as F_1 at the top K KPs ($K \in \{5, 10, 15\}$). Stemming is applied for computing F_1.

5.2 Baselines and Training Details

The first group for each K in Table 3 shows performance of the eight baseline unsupervised KPE approaches. We evaluate TextRank, SingleRank, TopicRank, MultipartiteRank, YAKE, EmbedRank using their implementations in the widely used toolkit PKE[5] with the default parameter settings. We evaluate SIFRank using the codebase [6] and the same parameters suggested by the authors (Sun et al., 2020). The original EmbedRank (Bennani-Smires et al., 2018) uses Sent2Vec for embedding and introduces embedding-based maximal marginal relevance (MMR) for improving diversity among selected KPs. For a fair comparison between the Phrase-Document method and our Document-Document MDERank, we design a new baseline EmbedRank(BERT) by replacing Sent2Vec with BERT and removing MMR.

We use YAKE (Campos et al., 2018) as θ to extract "keyphrases" for a document for KPEBERT pre-training, due to its high efficiency and consistent performance. Effects of the choice of θ on KPEBERT are analyzed in Appendix B where we compare YAKE and TextRank as θ. The number of pseudo labels for absolute and relative sampling for KPEBERT pre-training are 10 and 20, respectively. The λ is set to 0.1. The default parameter setting is the same as (Gao et al., 2021) except that we set the margin m for triplet loss to 0.2 and the learning rate to 3e-5. We use 4 NVIDIA V100 GPUs for training, the batch size is 2 per device and the gradient accumulation is 4. We train 10 epochs.

5.3 Performance Comparison

Table 3 shows F_1 at the top 5, 10, and 15 predictions from the baselines in the first group for each K and from our proposed MDERank(BERT) (default using BERT for embedding) and MDERank using KPEBERT for embedding, MDERank(KPEBERT), in the second group for each K. MDERank(BERT) and MDERank(KPEBERT) perform consistently well on all benchmarks. MDERank(BERT) outperforms EmbedRank(BERT) by 2.95 average F_1@15 and outperforms the previous SOTA SIFRank by 1.80 average F_1@15. MDERank further benefits from KPEBERT and overall MDERank(KPEBERT) achieves 3.53 average F_1@15 gain over SIFRank, especially on long-document datasets NUS and Krapivin. We also compute the average recalls of KPs with different phrase lengths (PL) in top-15 extracted KPs on all 6 bechmarks, for both EmbedRank(BERT) and MDERank(BERT), as shown in Table 4. We observe that EmbedRank has a strong bias for longer phrases, and PLs of its extracted phrases are concentrated in [2,3]. In contrast, PLs of KPs extracted by MDERank are more evenly distributed on diverse datasets. This analysis confirms that MDERank indeed alleviates the bias towards longer phrases from EmbedRank.

However, we observe that MDERank(BERT) has a large gap to SIFRank on DUC2001 and performs worse than EmbedRank(BERT) on Inspec. We investigate the reasons for these poorer performance. For DUC2001, different from other datasets collected from scientific publications, DUC2001 consists of open-domain news articles. The previous SOTA SIFRank introduces domain adaptation by combining weights from common corpus and domain corpus in the weight function of words for computing sentence embeddings, which may contribute significantly to its superior performance on

[2] https://huggingface.co/datasets/wikitext

[3] https://github.com/memray/OpenNMT-kpg-release

[4] https://github.com/sunyilgdx/SIFRank/tree/master/data

[5] https://github.com/boudinfl/pke

[6] https://github.com/sunyilgdx/SIFRank/tree/master

Method	EmbedRank(BERT)				MDERank(BERT)			
Data \ PL	1	2	3	>3	1	2	3	>3
Inspec	24.80	54.53	46.11	21.57	27.90	48.71	43.20	21.17
SemEval2017	24.91	53.68	48.05	9.84	37.28	47.07	43.99	9.76
SemEval2010	9.35	22.79	18.07	4.17	21.55	19.99	15.95	4.17
DUC2001	3.46	19.39	37.39	15.58	24.81	23.70	23.66	13.46
Krapivin	4.31	13.59	11.80	2.50	15.88	22.43	10.62	2.14
NUS	5.12	9.53	16.17	2.84	26.77	24.70	17.12	1.90

Table 4: The average recall of predicted KPs with different phrase lengths (PL) on all six benchmarks, from EmbedRank(BERT) and MDERank(BERT).

Method	Doc Len	F_1@5	F_1@10	F_1@15
EmbedRank(BERT)	128	8.76	14.75	16.28
	256	5.86	10.19	12.90
	512	3.75	6.34	8.11
MDERank(BERT)	128	12.86	16.06	16.67
	256	14.45	16.01	16.64
	512	15.24	18.33	17.95

Table 5: Effects of document lengths (the first 128, 256, 512 words of a document) on the KPE performance on the NUS dataset from EmbedRank(BERT) and MDERank(BERT).

DUC2001. As the default embedding model for MDERank, BERT is pre-trained on open-domain Wikipedia and Bookscorpus. However, as explained in Section 4, BERT does not emphasize learning significance of KPs or ranking between KPs. KPEBERT is designed to tackle this problem. Although the training data for KPEBERT, the open-domain WikiText language modeling dataset, is much smaller than English Wikipedia, with KPE-oriented representation learning in KPEBERT, the performance of MDERank(KPEBERT) improves remarkably and is comparable to SIFRank. For Inspec, the average PLs of gold labels of this dataset is relatively high (see Table 2). Also, on this dataset, when we move candidates with only 1 token to the end of ranking, MDERank(BERT) improves F_1@5, F_1@10, F_1@15 to 29.71 ,38.15, 39.46, an improvement of 3.54, 4.34 and 3.29, respectively. These analyses demonstrate that gold labels for Inspec are biased towards long PL. Therefore, EmbedRank with inductive bias for long PL may benefit from this annotation bias and perform well.

It is notable that MDERank particularly outperforms baselines on long-document datasets, verifying that MDERank could mitigate the weakness of performance degradation on long documents from PD methods. We further investigate effects of document length in Section 5.4. Absolute and relative sampling for KPEBERT achieve comparable performance on the 6 benchmarks with absolute sampling gaining a very small margin. Relative sampling performs better on NUS but is worse on Inspec and SemEval2017. We plan to continue exploring sampling approaches in future work, to reduce dependency on θ and improve KPEBERT.

5.4 Analyses

Effects of Document Length Section 5.3 demonstrates the superior performance of MDERank especially on long documents. We conduct two experiments to further analyze effects of document length on the performance of PD methods and MDERank. We choose EmbedRank(BERT) to represent PD methods. In the first experiment, both approaches use BERT for embedding and we truncate a document into the first 128, 256, 512 words. As shown in Table 5, F_1 for EmbedRank(BERT) drops drastically as the document length increases from 128 to 512, reflecting the weakness of EmbedRank(BERT) that the increased document length exacerbates discrepancy between sequence lengths of the document and KP candidates and mismatches between their embeddings, which degrades the KPE performance. In contrast, the performance of MDERank(BERT) improves steadily with increased document lengths, demonstrating the robustness of MDERank to document lengths and its capability to benefit KPE from more context in longer documents.

In the second experiment, we investigate effects of document length beyond 512 on EmbedRank and MDERank. To accommodate documents longer than 512, we choose BigBird (Zaheer et al., 2020) as the embedding model. BigBird replaces the full self-attention in Transformer with sparse attentions of global, local, and random attentions, reducing the quadratic complexity to sequence length from Transformer to linear. In order to select valid datasets for evaluation, we count the average percentage of gold label KPs appearing in the first m words in a document on the three longest datasets, DUC2001, NUS, and Krapivin. We observe that the first 500 words nearly cover 90% gold KPs in DUC2001, whereas 50% gold KPs in Krapivin are in the first 2500 words, and 50% gold KPs in NUS are in the first 2000 words. Therefore, we drop DUC2001 and use NUS and Krapivin for the second experiment. We keep the first 2500 and 2000 words for documents in Krapivin and NUS,

Method	NUS (512)			NUS (2000)			Krapivin (512)			Krapivin (2500)		
	F_1@5	F_1@10	F_1@15	F_1@5	F_1@10	F_1@15	F_1@5	F_1@10	F_1@15	F_1@5	F_1@10	F_1@15
EmbedRank(BERT)	3.75	6.34	8.11	–	–	–	4.05	6.60	7.84	–	–	–
EmbedRank(BigBird)	2.56	5.16	7.11	1.08	1.36	2.20	3.24	5.14	6.31	1.05	1.93	2.28
MDERank(BERT)	15.24	18.33	17.95	–	–	–	11.78	12.93	12.58	–	–	–
MDERank(BigBird)	15.42	17.68	17.81	15.36	19.56	20.33	11.62	11.99	11.70	11.33	12.71	12.70

Table 6: KPE performance from EmbedRank and MDERank using BERT for embedding (the first group) and the two approaches using BigBird for embedding (the second group). 512, 2000, 2500 in the parentheses represent the number of words kept for each document in datasets. The results for NUS(2000) and Krapivin (2500) are missing for EmbedRank(BERT) and MDERank(BERT) due to limitation on input sequence length from BERT.

Method	Pooling	Layer	DUC2001		
			F1@5	F1@10	F1@15
EmbedRank(BERT)	AvgPooling	3	16.19	21.21	22.12
		6	10.76	15.33	17.63
		12	10.41	15.15	17.69
	MaxPooling	3	6.97	11.04	12.27
		6	7.12	10.93	13.13
		12	8.12	11.62	13.58
MDERank(BERT)	AvgPooling	3	12.00	16.45	19.08
		6	12.40	17.07	19.02
		12	13.00	17.93	19.45
	MaxPooling	3	11.06	16.16	18.01
		6	11.06	15.91	17.98
		12	13.05	17.31	19.13

Table 7: KPE performance on DUC2001 from EmbedRank and MDERank using different BERT layers for embedding and pooling methods. AvgPooling and MaxPooling are employed on the output of a specific layer to produce document embeddings.

respectively. Table 6 demonstrates that on NUS, when increasing the document length from 512 to 2000, MDERank(BigBird) outperforms MDERank(BERT) by 2.52 F_1@15. On Krapivin, when increase the document length from 512 to 2500, MDERank also improves by 1.0 F_1@15. In contrast, the performance of EmbedRank degrades dramatically with longer context, since more context introduces more candidates into ranking and also worsens the discrepancy between lengths of document and phrases, which in turn greatly reduces the accuracy of similarity comparison.

Effects of Encoder Layers and Pooling Methods The findings in (Jawahar et al., 2019; Kim et al., 2020; Rogers et al., 2020) show that BERT captures a rich hierarchy of linguistic information, with surface features in lower layers, syntactic features in middle layers and semantic features in higher layers. We conduct experiments to understand the effects on MDERank and EmbedRank when using different BERT layers for embedding. We choose the third, the sixth, and the last layer from BERT-Base. We study the interactions between encoder layers and Max Pooling and Average Pooling methods. As shown in Table 7, for both AvgPooling and MaxPooling, F_1 from MDERank(BERT) shows a steady incline to the increase of layers. On the contrary, with AvgPooling, F_1 from EmbedRank(BERT) drastically drops as the layers rises from 3 to 12, probably due to that the lower BERT layer provides more rough and generic representations, which may alleviate mismatch in similarity comparison in Phrase-Document methods[7]. Compared to AvgPooling, MaxPooling produces weaker document embedding, which severely degrades the performance of EmbedRank and slightly degrades the performance of MDERank. On the other hand, MaxPooling probably reduces differences in embeddings across layers, hence performance of EmbedRank becomes stable across layers with MaxPooling. For both pooling methods, MDERank using the last BERT layer achieves the best results, which demonstrates that MDERank can fully benefit from stronger contextualized semantic representations.

6 Conclusion

We propose a novel embedding-based unsupervised KPE approach, MDERank, to improve reliability of similarity match compared to previous embedding-based methods. We also propose a novel self-supervised learning method and develop a KPE-oriented PLM, KPEBERT. Experiments demonstrate MDERank outperforms SOTA on diverse datasets and further benefits from KPEBERT. Analyses further verify the robustness of MDERank to different lengths of keyphrases and documents, and that MDERank benefits from longer context and stronger embedding models. Future work includes improving KPEBERT for MDERank by optimizing sampling strategies and pre-training methods.

[7]We also test the average F1@5, F1@10, F1@15 for EmbedRank(BERT) with AvgPooling and layer 3 on 6 datasets, which are 3.7, 1.8 and 1.6 absolute lower than MDERank(BERT).

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. *arXiv preprint arXiv:1704.02853*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. *arXiv preprint arXiv:1801.04470*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. *arXiv preprint arXiv:1803.08721*.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)*, pages 543–551.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *European Conference on Information Retrieval*, pages 806–810. Springer.

Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. 2019. An integrated approach for keyphrase generation via exploring the power of retrieval and extraction. *arXiv preprint arXiv:1904.03454*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Corina Florescu and Cornelia Caragea. 2017. 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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. Ptr: Prompt tuning with rules for text classification. *arXiv preprint arXiv:2105.11259*.

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.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

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.

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sanggoo Lee. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. *arXiv preprint arXiv:2002.00737*.

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

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Rui Meng, Xingdi Yuan, Tong Wang, Sanqiang Zhao, Adam Trischler, and Daqing He. 2020. An empirical study on neural keyphrase generation. *arXiv preprint arXiv:2009.10229*.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. *arXiv preprint arXiv:1704.06879*.

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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, pages 317–326. Springer.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.

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

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Dhruva Sahrawat, Debanjan Mahata, Mayank Kulkarni, Haimin Zhang, Rakesh Gosangi, Amanda Stent, Agniv Sharma, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. 2019. Keyphrase extraction from scholarly articles as sequence labeling using contextualized embeddings. *arXiv preprint arXiv:1910.08840*.

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.

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Appendices

A Effects of Masking Methods on MDERank

Given occurrences of a candidate KP c_i in a document d as $[p_1, p_2, \ldots, p_t]$, we study several methods to mask these occurrences and generate the masked document $d_M^{c_i}$, considering the potential bias e.g., frequency, sequence length, and nested phrases.

Mask Once The *Mask Once* method only masks the first occurrence of a candidate. This strategy eliminates the bias towards high frequency candidate KPs. However, it may prefer longer candidate KPs (i.e., candidate KPs that consist of more subwords) with the same argument shown in Section 1. MDERank may benefit from this masking strategy on datasets with annotation bias towards long keyphrases.

Mask Highest The *Mask Highest* method considers the collection of $d_M^{c_i}$s obtained by masking each occurrence of a candidate phrase c_i once in the document, and select the one that has the *smallest* cosine similarity with the embeddings of d. This method considers a balance of impacts from sequence length and frequency of candidate phrases.

Mask Subset One issue in KPE is that there may be heavy nesting among candidate KPs. For example, "support vector machine" may result in nested candidates such as "support vector machine", "support vector", "vector machine", and even "machine". Neither *Mask All* nor *Mask Once* strategy addresses this issue and hence the nested KPs may take up a large proportion in the final results, drastically damaging the diversity. We design the *Mask Subset* method to alleviate impact of nested candidate KPs. Firstly, all candidates are ranked by their phrase length in a descending order. Secondly, when generating a masked document for each candidate in order, *Mask Subset* records the positions of masked words and requires that each candidate could only be masked with words not in the recorded positions.

The KPE results from MDERank(BERT) using these masking strategies are shown in Table 8. The masking variants do not bring remarkable improvement compared with the results from Mask All, and Mask Once and Mask Highest perform even worse on the long-document datasets. This is because masking only one occurrence of a candidate will not emphasize the change of semantics significantly, especially on long documents. Mask subset could partially address the diversity problem by reducing the number of nested candidates selected by MDERank. Figure 3 shows a comparison on diversity between *Mask Subset* and other methods, where the evaluation metric for diversity is defined in Equation 1. The Phrase-Document method refers to EmbedRank(BERT). We could see from Figure 3 that MDERank with Mask Subset indeed boosts the diversity over Mask All and even exceeds gold labels on several datasets.

$$Diveristy(d) = \frac{t_u}{t_n} * 100 \quad (1)$$

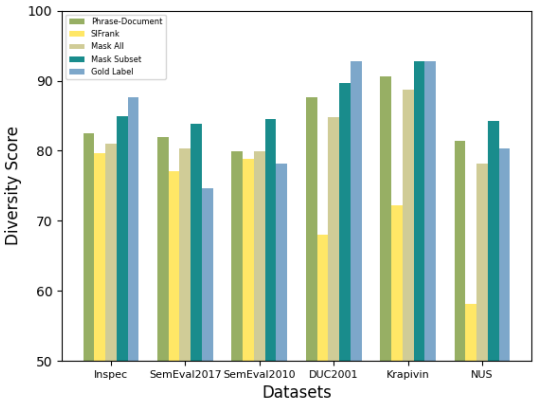

Figure 3: Diversity scores from different methods on various datasets. A higher bar indicates a better diversity. The diversity of gold keyphrases are in blue and on the right.

B Effects of the Choice of θ on KPEBERT

We also investigate the effects of the choice of θ on KPEBERT. We explore alternative unsupervised KPE methods as θ for generating pseudo labels for KPEBERT pre-training. Besides YAKE, when balancing the extraction speed and KPE quality, TextRank is another choice for θ. As shown in Table 3, YAKE performs better than TextRank on long-document datasets but worse on short-document datasets. After replacing YAKE with TextRank as θ for producing pseudo labels and training KPEBERT, the KPE results of the respective MDERank(KPEBERT) are shown in Table 9. We observe that MDERank(KPEBERT) using YAKE as θ significantly outperforms MDERank(KPEBERT) using TextRank as θ, on both short-document datasets

F_1@K	Method	Dataset						
		Inspec	SemEval2017	SemEval2010	DUC2001	Krapivin	NUS	AVG
5	Mask All	26.17	22.81	12.95	13.05	11.78	15.24	17.00
	Mask Once	27.93	20.56	10.16	9.11	4.61	3.92	12.72
	Mask Highest	27.93	20.56	10.16	9.11	4.65	3.92	12.72
	Mask Subset	29.25	21.50	10.26	12.05	8.50	9.61	15.20
10	Mask All	33.81	32.51	17.07	17.31	12.93	18.33	21.99
	Mask Once	37.38	30.95	15.40	13.49	7.21	6.52	18.49
	Mask Highest	37.42	30.97	15.32	13.46	7.24	6.56	18.50
	Mask Subset	36.55	31.30	15.88	16.73	9.99	13.43	20.65
15	Mask All	36.17	37.18	20.09	19.13	12.58	17.95	23.85
	Mask Once	39.11	36.07	17.69	16.47	8.15	8.85	21.06
	Mask Highest	39.36	36.10	17.76	16.45	8.20	8.85	21.12
	Mask Subset	38.08	36.67	17.83	19.19	10.48	14.65	22.82

Table 8: F_1@K ($K \in \{5, 10, 15\}$) from MDERank(BERT) using different masking methods, where Mask All refers to the masking method described in Section 3.

F_1@K	θ	Dataset						
		Inspec	SemEval2017	SemEval2010	DUC2001	Krapivin	NUS	AVG
5	TextRank	28.93	21.34	11.46	13.30	7.85	7.57	15.08
	YAKE	28.06	21.63	12.95	22.51	12.91	14.11	18.70
10	TextRank	38.13	32.71	17.23	19.15	10.47	10.59	21.38
	YAKE	35.80	32.23	17.95	26.97	14.36	17.72	24.17
15	TextRank	39.49	37.95	19.89	22.11	11.40	12.83	23.95
	YAKE	37.43	37.52	20.69	26.28	13.58	17.95	25.58

Table 9: The KPE performance (F_1@K) from MDERank(KPEBERT) with KPEBERT pre-trained using YAKE and TextRank as θ for producing pseudo labels, respectively. **AVG** is the average F1@K on all six benchmarks

Method	F1@K	Dataset						
		Inspec	SemEval2017	SemEval2010	DUC2001	Krapivin	NUS	AVG
EmbedRank(Cos)	5	28.92	20.03	10.46	8.12	4.05	3.75	12.56
	10	38.55	31.01	16.35	11.62	6.60	6.34	18.41
	15	39.77	36.72	19.35	13.58	7.84	8.11	20.90
EmbedRank(Euc)	5	29.28	19.77	9.47	7.92	4.13	4.04	12.44
	10	38.23	30.58	16.35	11.61	6.66	6.52	18.33
	15	39.80	36.14	19.02	13.49	7.71	**8.18**	20.72
MDERank(Cos)	5	26.17	22.81	12.95	13.05	11.78	15.07	16.97
	10	33.81	32.51	17.07	17.31	12.93	19.20	22.14
	15	36.17	37.18	19.02	19.13	12.58	**19.62**	23.95
MDERank(Euc)	5	26.25	22.83	12.76	13.10	11.29	15.24	16.91
	10	33.83	32.59	17.15	17.45	12.15	18.29	21.91
	15	36.25	37.24	20.22	19.33	11.82	18.02	23.81

Table 10: The KPE performance from MDERank and EmbedRank using Cosine and Euclidean as similarity measure, where EmbedRank is EmbedRank(BERT) as in Section 5.2 and MDERank is MDERank(BERT).

and long-document datasets (except on Inspec). Although on average YAKE performs worse than TextRank on the six benchmarks, the better performance from YAKE on long documents coupled with its consistent performance may be a crucial factor when choosing θ for pre-training KPEBERT. Results in Table 3 shows that MDERank(KPEBERT) yields superior performance on both short and long documents. In other words, KPEBERT benefits from the stable performance from YAKE on long documents for pseudo labeling, and KPEBERT also exhibits robustness to the relatively low performance on short documents from YAKE.

Impact of Similarity Measure The common similarity measures include Cosine and Euclidean distance. However, the choice of similarity measure does not matter for MDERank performance.

We conduct experiments to investigate the impact of the similarity measure on the performance of MDERank, and the results are shown in Table 10. We observe that Cosine and Euclidean similarity measure are not a salient factor for the ranking results for both EmbedRank(BERT) and MDERank(BERT).