Unsupervised Keyphrase Extraction by Learning Neural Keyphrase Set Function

Mingyang Song[♠], **Haiyun Jiang**[♣*], **Lemao Liu**[♣], **Shuming Shi**[♣], **Liping Jing**[♠*]
[♣]Tencent AI Lab, Shenzhen, China
[♠]Beijing Key Lab of Traffic Data Analysis and Mining
[♠]Beijing Jiaotong University, Beijing, China
mingyang.song@bjtu.edu.cn

Abstract

We create a *paradigm shift* concerning building unsupervised keyphrase extraction systems in this paper. Instead of modeling the relevance between an individual candidate phrase and the document as in the commonly used framework, we formulate the unsupervised keyphrase extraction task as a document-set matching problem from *a set-wise perspective*, in which the document and the candidate set are globally matched in the semantic space to particularly take into account the interactions among all candidate phrases. Since it is intractable to exactly extract the optimal subset by the document-set matching function during the inference, we propose an approximate approach, which obtains the candidate subsets via a set extractor agent learned by reinforcement learning. Exhaustive experimental results demonstrate the effectiveness of our model, which outperforms the recent state-of-the-art unsupervised keyphrase extraction baselines by a large margin.

1 Introduction

Keyphrase Extraction (KE) is the task of extracting a keyphrase set that provides readers with high-level information about the key ideas or important topics described in the document. KE methods can be divided into supervised (Sun et al., 2021; Song et al., 2021, 2022a) or unsupervised (Bennani-Smires et al., 2018; Sun et al., 2020). The former requires large-scale annotated training data and is often domain-specific, whereas unsupervised methods do not need annotated data (Hasan and Ng, 2014). Therefore, in this paper, we focus on Unsupervised Keyphrase Extraction (UKE).

Currently, most UKE methods mainly consist of two components: candidate set generation and keyphrase importance estimation. The former uses heuristic rules to obtain a candidate set for a given document. The latter scores individual phrase from the candidate set with respect to a document, and then selects top-ranked phrases to form a keyphrase set. For example, Bennani-Smires et al. (2018); Sun et al. (2020); Liang et al. (2021); Song et al. (2022b); Zhang et al. (2022) address it with the pre-trained embeddings (Peters et al., 2018; Devlin et al., 2019). These methods independently estimate the relevance between each phrase in the candidate set and the document as the importance of the phrase from a point-wise perspective, as illustrated in Figure 1(a).

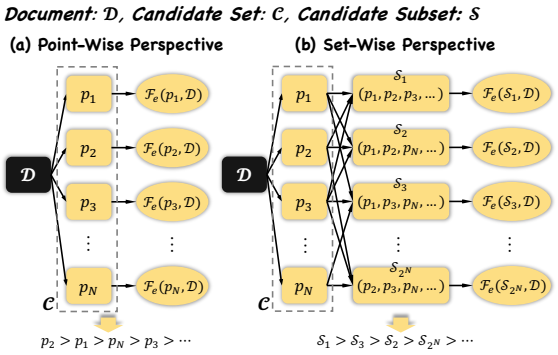

Figure 1: Illustration of extracting keyphrases from different perspectives. $\mathcal{F}_s(\cdot)$ denotes a scoring function measuring the relevance between a candidate phrase p_i (a) or a candidate subset $\mathcal{S}_i$ (b) and the document $\mathcal{D}$.

Unfortunately, the above point-wise models are essentially phrase-level UKE approaches, and they can not take into account the interactions among all candidate phrases and fails to consider the semantics of the complete candidate set. This makes them more inclined to select keyphrases with high-frequency words while ignoring the coupling of multiple phrases. As a result, the diversity of the selected keyphrases suffers as quantified in our experiments (as shown in Table 6), leading to suboptimal performance.

To address the above issue, we investigate extracting keyphrases globally from a set-wise perspective (as illustrated in Figure 1(b)) and concep-

*Corresponding Author

Findings of the Association for Computational Linguistics: ACL 2023, pages 2482–2494
July 9-14, 2023 ©2023 Association for Computational Linguistics

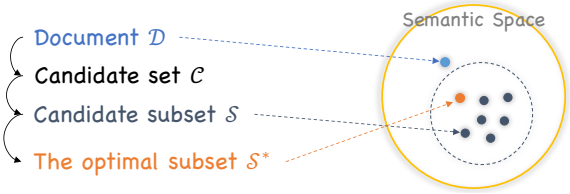

Figure 2: The document-set matching framework. Intuitively, better *candidate subset* should be semantically closer to *the document* in the semantic space, while *the optimal subset* should be the closest.

tualize the UKE task as a document-set matching problem, as shown in Figure 2. Specifically, the proposed UKE system is based on a document-set matching framework as the set function that measures the relevance between a candidate set and its corresponding document in the semantic space via a siamese-based neural network. The set function is learned by the margin-based triplet loss with orthogonal regularization, effectively capturing similarities of documents and candidate sets. However, it is intractable to exactly search the optimal subset from the candidate set by the set function during the inference because the subset space is exponentially large, and the set function is non-decomposable. To this end, we propose an approximate method whose key idea is to learn a set extractor agent and search for efficient inference. Concretely, after the neural keyphrase set function is well-trained, we use it to calculate the document-set matching score as the reward. Then, we adopt the policy gradient training strategy to train the set extractor agent for extracting the optimal subset with the highest reward from numerous candidate subsets. Ideally, the optimal subset is the closest semantically to the document, as shown in Figure 2. Exhaustive experiments demonstrate the effectiveness of our model SetMatch: it effectively covers the ground-truth keyphrases and obtains higher recall than the traditional heuristics, and it outperforms recent strong UKE baselines.

We summarize our contributions as follows:

- Instead of individually scoring each phrase, we formulate the UKE task as a document-set matching problem and propose a novel set-wise framework.
- Since the exact search with the document-set matching function, we propose an approximate method by learning a set extractor agent to search the keyphrase set.

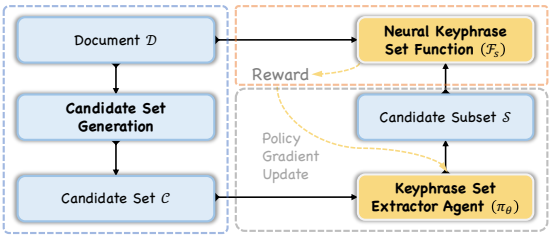

Figure 3: The overall pipeline of our model.

- Experiments show that it has achieved superior performance compared with the state-of-the-art UKE baselines on three benchmarks.

2 Methodology Overview

In this paper, keyphrases are *globally* selected from *a set-wise perspective*. More formally, consider a KE system: given the document $\mathcal{D}$, generate its candidate set first. And then, an optimal subset $\mathcal{S}^* \subseteq \mathcal{C}$ is selected from the candidate set $\mathcal{C}$. To achieve this goal, we propose a two-stage model (SetMatch), including candidate set generation and neural keyphrase set function $\mathcal{F}_s$. First, candidate set generation aims to generate a candidate set $\mathcal{C}$ from the document $\mathcal{D}$ with a *higher recall* to cover more ground-truth keyphrases (Sec 2.1). Second, a neural keyphrase set function $\mathcal{F}_s$ is learned to estimate the document-set matching score (Sec 2.2), which is used to guide the keyphrase set extractor agent to search an optimal subset (Sec 2.3).

2.1 Candidate Set Generation

We adopt various strategies to obtain a candidate set to cover the ground-truth keyphrases fully. These strategies can be mainly divided into two categories, using heuristic rules and pre-trained language models (fine-tuned via keyphrase extraction or generation tasks). The former first tokenize the document, tag the document with part-of-speech tags, and extract candidate phrases based on part-of-speech tags. Next, only keep noun phrases that consist of zero or more adjectives followed by one or multiple nouns. The latter uses neural keyphrase extraction or generation models based on Pre-trained Language Models (PLMs) fine-tuning on other corpora. The details are described in Sec 5.

2.2 Neural Keyphrase Set Function

To estimate the importance from a set-wise perspective, we propose a novel neural keyphrase set function $\mathcal{F}_s$, which is implemented by a document-set matching framework (Sec 3). With the neural

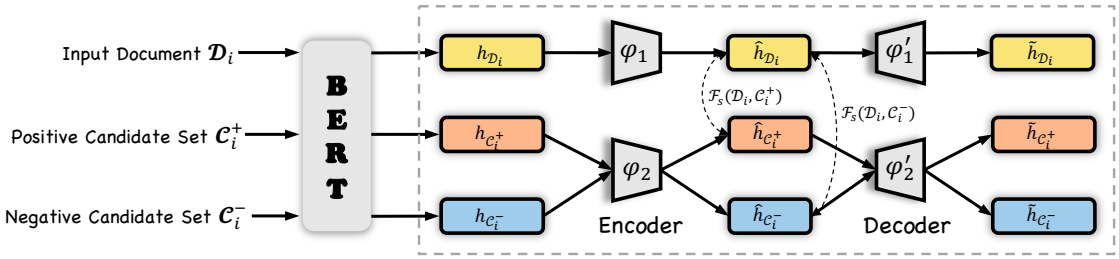

Figure 4: The overall architecture of our document-set matching module.

keyphrase set function $\mathcal{F}_s$, we can score all candidate subsets in the candidate set $\mathcal{C}$ and thus find the optimal subset $\mathcal{S}^*$ depending on these scores.

2.3 Keyphrase Set Extractor Agent

However, it is intractable to exactly search an optimal subset by the keyphrase set function $\mathcal{F}_s$ during the inference because the subset space is exponentially large, and the keyphrase set function $\mathcal{F}_s$ is non-decomposable. Therefore, we propose a keyphrase set extractor agent to search the optimal subset $\mathcal{S}^*$, which is trained by using the keyphrase set function $\mathcal{F}_s$ as the reward via the policy gradient training strategy to select the optimal subset $\mathcal{S}^*$ as the keyphrases (Sec 4). Finally, we infer the optimal subset by using the learned set extractor agent rather than $\mathcal{F}_s$.

3 Neural Keyphrase Set Function $(\mathcal{F}_s)$

There are many ways to judge whether a keyphrase set is good or bad under the document $\mathcal{D}$. One intuitive way is through a matching framework. Therefore, we formulate the neural keyphrase set function $\mathcal{F}_s$ as a document-set matching task in which the document $\mathcal{D}$ and the candidate set $\mathcal{C}$ will be matched in a semantic space, as shown in Figure 2. Then, we propose a margin-based triplet loss with multiple perspectives orthogonal regularization $\mathcal{L}_\mathcal{E}$ to optimize the Siamese-BERT Auto-Encoder architecture. The following section details how we instantiate our neural keyphrase set function $\mathcal{F}_s$ using a simple siamese-based architecture.

3.1 Siamese-BERT Auto-Encoder

Inspired by siamese network structure (Bromley et al., 1993), we construct a Siamese-BERT Auto-Encoder architecture to match the document $\mathcal{D}$ and the candidate set $\mathcal{C}$. Concretely, our Siamese-BERT Auto-Encoder consists of two BERTs with shared weights, two auto-encoders, and a cosine-similarity layer to predict the document-set score. The overall architecture is shown in Figure 4.

Given a batch of candidate sets $\{\mathcal{C}_i\}_{i=1}^M$ and documents $\{\mathcal{D}_i\}_{i=1}^M$, we adopt the original BERT (Devlin et al., 2019) to derive the semantically meaningful embeddings as follows,

$$h_{\mathcal{C}_i} = \text{BERT}(\mathcal{C}_i), h_{\mathcal{D}_i} = \text{BERT}(\mathcal{D}_i), \quad (1)$$

where M indicates the batch size. $h_{\mathcal{C}_i}, h_{\mathcal{D}_i} \in \mathbb{R}^{d_r}$ are the i-th candidate set $\mathcal{C}_i$ and document $\mathcal{D}_i$ representations within a training batch. Here, we use the vector of the '[CLS]' token from the top BERT layer as the representation of the candidate set $\mathcal{C}$ and the document $\mathcal{D}$. Next, we employ two auto-encoders (with two encoders φ_1, φ_2 and two decoders φ'_1, φ'_2, as shown in Figure 4) to transfer BERT representations into the latent space as,

$$\begin{aligned} \hat{h}_{\mathcal{D}_i} &= \varphi_1(h_{\mathcal{D}_i}), \hat{h}_{\mathcal{C}_i} = \varphi_2(h_{\mathcal{C}_i}), \\ \tilde{h}_{\mathcal{D}_i} &= \varphi'_1(\hat{h}_{\mathcal{D}_i}), \tilde{h}_{\mathcal{C}_i} = \varphi'_2(\hat{h}_{\mathcal{C}_i}), \end{aligned} \quad (2)$$

where $\varphi_1, \varphi_2 \in \mathbb{R}^{d_r \times d_l}$ and $\varphi'_1, \varphi'_2 \in \mathbb{R}^{d_l \times d_r}$ are learnable parameters. Here, let $\hat{h}_{\mathcal{C}_i}, \hat{h}_{\mathcal{D}_i} \in \mathbb{R}^{d_l}$ denote the representations of the candidate set $\mathcal{C}_i$ and the document $\mathcal{D}_i$ in the latent space, respectively. Finally, their similarity score is measured by $\mathcal{F}_s(\mathcal{C}_i, \mathcal{D}_i) = \text{cosine}(\hat{h}_{\mathcal{C}_i}, \hat{h}_{\mathcal{D}_i})$.

3.2 Margin-based Triplet Loss with Orthogonal Regularization

To fine-tune Siamese-BERT Auto-Encoder, we use a margin-based triplet loss with orthogonal regularization to update the weights. We use a simple and intuitive way to generate positive $\mathcal{C}_i^+$ and negative $\mathcal{C}_i^-$ candidate sets. Most existing embedding-based UKE models (Liang et al., 2021; Ding and Luo, 2021) truncate the document to satisfy the encoding requirements of BERT. However, truncating documents will lose a small number of phrases, thus reducing the recall of the candidate set $\mathcal{C}$. Therefore, we generate a positive candidate set $\mathcal{C}_i^+$ (i.e.,

$\mathcal{A}_1^{\dagger}$, as illustrated in Table 3) before truncating the document $\mathcal{D}$, and generate a negative candidate set $\mathcal{C}_i^-$ (i.e., $\mathcal{A}_1$, as illustrated in Table 3) after truncating the document $\mathcal{D}$ (more details in Sec 5). Then, the loss $\mathcal{L}_{\mathcal{T}}$ can be computed as,

$$\mathcal{L}_{\mathcal{T}} = \sum_{i=1}^{M} \max(\mathcal{F}_s(\hat{h}_{\mathcal{D}_i}, \hat{h}_{\mathcal{C}_i^-}) - \mathcal{F}_s(\hat{h}_{\mathcal{D}_i}, \hat{h}_{\mathcal{C}_i^+}) + \delta, 0), \quad (3)$$

where δ denotes the margin. The basic idea of $\mathcal{L}_{\mathcal{T}}$ is to let the positive candidate set with higher recall have a higher document-set matching score than the negative candidate set with lower recall. Furthermore, we propose orthogonal regularization from multiple perspectives, which explicitly encourages each representation within a batch to be different from the other. This is inspired by Bousmalis et al. (2016), who adopts orthogonal regularization to encourage representations across domains to be as distinct as possible. Here, we use the following equations as the orthogonal regularization:

$$\begin{aligned}
\mathcal{L}_{\mathcal{CC}} &= \sum_{i=1}^{M} \sum_{j,j\neq i} \mathcal{F}_s(\hat{h}_{\mathcal{C}_i}, \hat{h}_{\mathcal{C}_j})^2, \\
\mathcal{L}_{\mathcal{DD}} &= \sum_{i=1}^{M} \sum_{j,j\neq i} \mathcal{F}_s(\hat{h}_{\mathcal{D}_i}, \hat{h}_{\mathcal{D}_j})^2, \\
\mathcal{L}_{\mathcal{CD}} &= \sum_{i=1}^{M} \sum_{j,j\neq i} \mathcal{F}_s(\hat{h}_{\mathcal{C}_i}, \hat{h}_{\mathcal{D}_j})^2,
\end{aligned} \quad (4)$$

where $\mathcal{L}_{\mathcal{CC}}$ encourages the similarities between all candidate sets under a batch as distinct as possible, $\mathcal{L}_{\mathcal{DD}}$ encourages the similarities between all documents under a batch as distinct as possible, $\mathcal{L}_{\mathcal{CD}}$ encourages the similarities between candidate sets and documents under a batch as distinct as possible. Therefore, the final loss function $\mathcal{L}_{\mathcal{E}}$ of the neural keyphrase set function is re-formulated as,

$$\mathcal{L}_{\mathcal{E}} = \lambda_1 \mathcal{L}_{\mathcal{T}} + \lambda_2(\mathcal{L}_{\mathcal{CC}} + \mathcal{L}_{\mathcal{DD}} + \mathcal{L}_{\mathcal{CD}}) + \lambda_3(\mathcal{L}_{\mathcal{D}} + \mathcal{L}_{\mathcal{C}}) \quad (5)$$

where $\lambda_1, \lambda_2, \lambda_3$ are the balance factors. Here, $\mathcal{L}_{\mathcal{D}}, \mathcal{L}_{\mathcal{C}}$ denote the reconstruction loss of our two auto-encoders and are calculated as follows,

$$\mathcal{L}_{\mathcal{D}} = ||h_{\mathcal{D}_i} - \tilde{h}_{\mathcal{D}_i}||_2, \mathcal{L}_{\mathcal{C}} = ||h_{\mathcal{C}_i} - \tilde{h}_{\mathcal{C}_i}||_2, \quad (6)$$

where $||\mathcal{X}||_2$ indicates L_2 norm of each element in a matrix $\mathcal{X}$. After the set function $\mathcal{F}_s$ is well-trained, we fix its parameters and only use it as a non-differential metric to measure a document-set matching score without optimizing parameters.

4 Keyphrase Set Extractor Agent

As mentioned before, it is intractable to search the optimal subset by the set function precisely. Therefore, we propose a keyphrase set extractor agent to efficiently search an optimal subset. We first exploit a pre-trained BERT model to obtain representations of phrases in the candidate set $\mathcal{C}$ and the document $\mathcal{D}$, and then learn a subset sampling network to sample a subset $\mathcal{S}$ from the candidate set $\mathcal{C}$ based on their representations. After obtaining the candidate subset $\mathcal{S}$, we use the keyphrase set function $\mathcal{F}_s$ to calculate the document-set matching score $\mathcal{F}_s(\mathcal{S}, \mathcal{D})$ as the reward $\mathcal{R}(\mathcal{S}, \mathcal{D})$ to optimize the keyphrase set extractor agent for extracting an optimal subset $\mathcal{S}^*$ via reinforcement learning.

4.1 Encoding Network

We employ a pre-trained BERT model to obtain $\mathcal{H}, h_{\mathcal{D}}$, the representations of phrases in the candidate set $\mathcal{C}$ and the document $\mathcal{D}$, respectively. Here, the representations are obtained by using average pooling on the output of the last BERT layer:

$$\begin{aligned}
h_{\mathcal{D}} &= \text{BERT}(\mathcal{D}), \\
\mathcal{H} &= [h_{p_1}^{\top}, ..., h_{p_n}^{\top}, ..., h_{p_N}^{\top}]^{\top} \\
&= \text{BERT}(p_i), i = 1, ..., n, ..., N \ ,
\end{aligned} \quad (7)$$

where $h_{\mathcal{D}}$ denotes the document representation and h_{p_n} is the n-th phrase representation in the candidate set $\mathcal{C}$ (contains N candidate phrases).

4.2 Candidate Subset Searching

To obtain a candidate subset $\mathcal{S}$ from the candidate set $\mathcal{C}$, we adopt a self-attention layer as the extractor network to search subsets. We calculate the attention function on all candidate phrases in the candidate set $\mathcal{C}$ simultaneously, packed together into a matrix $\mathcal{H}$. We compute the matrix of outputs as follow,

$$\hat{\mathcal{H}} = \mathcal{H}\mathbf{W}_1 + \mathit{REP}(h_{\mathcal{D}})\mathbf{W}_2, \quad (8)$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d_r \times d_r}$ are the trainable parameters and the *REP* operator converts the input vector to a $\mathbb{R}^{N \times d_r}$ matrix by repeating elements up to N rows. Then, the probability distribution can be obtained by,

$$\pi_\theta(\mathcal{S}, \mathcal{D}) = \prod_{p \in \mathcal{S}} \text{softmax}(f_d(\hat{\mathcal{H}}))[p] \quad (9)$$

where $\pi_\theta(\mathcal{S}, \mathcal{D})$ denotes the predicted probability over the candidate set $\mathcal{C}$, θ indicates the trainable parameters of our keyphrase set extractor, $f_d \in \mathbb{R}^{d_r \times 1}$

Dataset	# Doc.	Type	Avg. # Words	Avg. # Keyphrases	Present Keyphrases *in Truncated Doc. (512)*	Present Keyphrases *in Original Doc.*
Inspec (Hulth, 2003)	500	*Short*	134.60	9.83	**0.7341**	
DUC2001 (Wan and Xiao, 2008)	308	*Long*	847.24	8.08	**0.8436**	**0.9339** (↑0.0903)
SemEval2010 (Kim et al., 2010)	100	*Long*	1587.52	12.04	**0.5156**	**0.6576** (↑0.1420)

Table 1: The statistics of several benchmarks. **# Doc.** is the number of documents. **Type** indicates the length of documents. **Avg. # Words** is the average number of words for documents. **Present Keyphrases *in Truncated Doc. (512)*** and ***in Original Doc.*** indicate the ratio of keyphrases, which present in the truncated and original documents.

is a fully-connected layer, and p is the candidate phrase in the candidate set $\mathcal{C}$. To obtain the candidate subset, we rank phrases in the candidate set $\mathcal{C}$ with the predicted probability $\pi_\theta(\mathcal{S}, \mathcal{D})$ and extract top-ranked $K(K < N)$ keyphrases as a candidate subset $\mathcal{S}$.

4.3 Reinforce-Guided Selection

We exploit an exploitation and exploration training strategy to train the set extractor agent for optimizing its parameters. Here, we adopt the policy gradient algorithm (REINFORCE, (Williams, 1992)) to optimize the policy $\pi_\theta(\mathcal{S}, \mathcal{D})$. Specifically, in a training iteration, we first use the policy $\pi_\theta(\mathcal{S}, \mathcal{D})$ to search a candidate subset $\mathcal{S}$ from the candidate set $\mathcal{C}$ of the document $\mathcal{D}$. Next, the well-trained set function $\mathcal{F}_s$ computes a document-set matching score $\mathcal{F}_s(\mathcal{S}, \mathcal{D})$ between the candidate subset $\mathcal{S}$ and the document $\mathcal{D}$. Finally, we treat the document-set matching score $\mathcal{F}_s(\mathcal{S}, \mathcal{D})$ as the reward $\mathcal{R}(\mathcal{S}, \mathcal{D})$ to optimize the policy $\pi_\theta(\mathcal{S}, \mathcal{D})$ with the policy gradient :

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(\mathcal{S}, \mathcal{D}) \mathcal{R}(\mathcal{S}, \mathcal{D})]. \quad (10)$$

Inspired by the self-critical training strategy (Rennie et al., 2017), we propose a new teacher-critical training strategy to regularize the reward $\mathcal{R}(\mathcal{S}, \mathcal{D})$, which uses the top-K predicted keyphrases of the baselines (e.g., JointGL (Liang et al., 2021)) as a reference set $\hat{\mathcal{S}}$. Ideally, when maximizing rewards, the teacher-critical training strategy ensures that our model obtains an optimal candidate subset $\mathcal{S}^*$ better than the reference set $\hat{\mathcal{S}}$. Then, we calculate a document-set matching score $\mathcal{F}_s(\hat{\mathcal{S}}, \mathcal{D})$ to regularize the reward $\mathcal{R}(\mathcal{S}, \mathcal{D})$. Finally, the expected gradient can be approximated by,

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(\mathcal{S}, \mathcal{D})(\mathcal{R}(\mathcal{S}, \mathcal{D}) - \mathcal{F}_s(\hat{\mathcal{S}}, \mathcal{D}))]. \quad (11)$$

Generally, the policy $\pi_\theta(\mathcal{S}, \mathcal{D})$ is gradually optimized through the continuous iteration of the training process to search a better candidate subset $\mathcal{S}$ to obtain a higher reward $\mathcal{R}(\mathcal{S}, \mathcal{D})$. The candidate subset $\mathcal{S}^*$ with the highest reward $\mathcal{R}(\mathcal{S}^*, \mathcal{D})$ is the final predicted keyphrase set of the document $\mathcal{D}$.

5 Experiments

5.1 Datasets and Evaluation Metrics

We verify our model on three benchmarks, including the DUC2001 (Wan and Xiao, 2008), Inspec (Hulth, 2003), and SemEval2010 (Kim et al., 2010) datasets. Both keyphrases and their corresponding document are preprocessed via Porter Stemmer[1]. The statistics are provided in Table 1.

Following the recent studies (Liang et al., 2021; Ding and Luo, 2021; Zhang et al., 2022), the performance of our model SetMatch and the selected baselines is evaluated using Precision (P), Recall (R), and F1 measure (F1) on the top 5, 10, and 15 ranked phrases.

5.2 Baselines

We compare the proposed model with recent state-of-the-art UKE baselines, which extract keyphrases from a point-wise perspective (KeyGames (Saxena et al., 2020), EmbedRankd2v, EmbedRanks2v (Bennani-Smires et al., 2018), SIFRank, SIFRank+ (Sun et al., 2020), JointGL (Liang et al., 2021), MDERank (Zhang et al., 2022)).

5.3 Implementation Details

Candidate Set Generation. All the models use Stanford CoreNLP Tools[2] for tokenizing, part-of-speech tagging and noun phrase chunking. Three regular expressions are used to extract noun phrases as the candidate set via the python package NLTK[3]: $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$, as shown in Table 3. Furthermore, we use two fine-tuned pre-trained language models

[1] https://tartarus.org/martin/PorterStemmer/
[2] https://stanfordnlp.github.io/CoreNLP/
[3] https://github.com/nltk

Embedding-based UKE Model	DUC2001			Inspec			SemEval2010		
	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15
Point-Wise Perspective									
EmbedRankd2v (Bennani-Smires et al., 2018)	24.02	28.12	28.82	31.51	37.94	37.96	3.02	5.08	7.23
EmbedRanks2v (Bennani-Smires et al., 2018)	27.16	31.85	31.52	29.88	37.09	38.40	5.40	8.91	10.06
KeyGames (Saxena et al., 2020)	24.42	28.28	29.77	32.12	40.48	40.94	11.93	14.35	14.62
SIFRank (Sun et al., 2020)	24.27	27.43	27.86	29.11	38.80	39.59	-	-	-
SIFRank+ (Sun et al., 2020)	30.88	33.37	32.24	28.49	36.77	38.82	-	-	-
JointGL(Liang et al., 2021)	28.62	35.52	36.29	32.61	40.17	41.09	13.02	19.35	21.72
MDERank (Zhang et al., 2022)	23.31	26.65	26.42	27.85	34.36	36.40	13.05	18.27	20.35
Set-Wise Perspective									
SetMatch	**31.19**	**36.34**	**38.72**	**33.54**	**40.63**	**42.11**	**14.44**	**20.79**	**24.18**

Table 2: Performance of embedding-based UKE models on the DUC2001, Inspec, and SemEval2010 test sets in terms of F1@5, F1@10, and F1@15 evaluation metrics. The best/2nd-best scores are in bold/underlined.

($\mathcal{B}_1$[4] and $\mathcal{B}_2$[5], as shown in Table 3) to generate candidate sets. Here, we take the entire document as input for the truncated document to generate a candidate set (*document-level*). For the document, without truncating, we leverage fine-tuned PLMs to obtain candidate keyphrases from each sentence in the document individually and combine them as a candidate set (*sentence-level*).

Neural Keyphrase Set Function. Specifically, we set the margin δ for the margin-based triplet loss to 1, $\lambda_1 = \lambda_2 = \lambda_3 = 1/3$, and the learning rate is set to $5e$-5 for both the neural keyphrase set function and the keyphrase set extractor agent. We use a single NVIDIA A4000 GPU for training, the batch size is 2. We train twenty epochs. $d_r = 768$ and $d_l = 512$. We set K to 15 and N to 30. In this paper, we use $\mathcal{A}_1^\dagger \cup \mathcal{B}_2^\dagger$, $\mathcal{A}_1 \cup \mathcal{B}_2$, and $\mathcal{A}_1 \cup \mathcal{B}_1$ to obtain candidate sets for the Inspec, DUC2001, and SemEval2010 datasets, respectively.

Candidate Set Pruning. The subset sampling idea of our subset sampler is more intuitive, while it suffers from combinatorial explosion problems. For example, how could we determine the number of phrases in the candidate set, or should we score all possible subsets? To alleviate these difficulties, we propose a simple candidate pruning strategy, which adopts the recent baseline JointGL (Liang et al., 2021) to prune the candidate set from a point-wise perspective and keep top-ranked N phrases as the candidate set $\mathcal{C}$.

5.4 Results and Analysis

Table 2 illustrates the experimental results on the DUC2001, Inspec, and SemEval2010 datasets.

Analysis. The experimental results show that globally extracting keyphrases from a set-wise perspective helps our model outperform recent state-of-the-art baselines across the benchmark datasets. The detailed analysis is presented as follows:

(1) The keyphrases of the document are usually considered to be disordered and treated as a set. Similar claims have been reported previously in the keyphrase generation literature (Ye et al., 2021; Xie et al., 2022). However, most UKE models score and extract keyphrases from a point-wise perspective, which also rank good keyphrases in order. The impact caused by ranking in order is also visible in the results. It will result in higher scores for F@5 and F@10 but less boost for F@15. Instead, our model globally extracts keyphrases from the set-wise perspective. Not only does it focus on modeling the relationship between phrases within the document at a deeper level, but it also ensures that the extracted keyphrase set is semantically closer to its corresponding document in the semantic space. Moreover, the keyphrases predicted by our model are disordered.

(2) Most existing embedding-based UKE models obtain the candidate set and the embeddings of phrases after truncating the document. Notable that this is done for two main reasons. First, it benefits to calculate the document-phrase matching similarity. Second, it is subject to the limitation of the input length by the pre-trained language model. However, truncating documents reduces the quality of candidate sets, reducing the performance of keyphrase extraction. Our document-set matching framework alleviates this problem, allowing our model to consider all phrases in the original document to form a candidate set. From the results, the improvement of our model on the DUC2001

[4] https://github.com/Shivanandroy/KeyPhraseTransformer

[5] https://github.com/MaartenGr/KeyBERT

Candidate Set Generation Strategy	Inspec		DUC2001		SemEval2010	
	R@50	**R@M**	R@50	**R@M**	R@50	**R@M**
*Regular Expression for **Truncated** Document (with length limitation→512)*						
$\mathcal{A}_1 \rightarrow \{< NN.* \mid JJ > * < NN.* >\}$	0.5350	0.5359$_{(26)}$	0.5845	0.6840$_{(76)}$	0.2885	0.3301$_{(69)}$
$\mathcal{A}_2 \rightarrow \{< JJ \mid VBG > * < NN.* > 0, 3\}$	0.5267	0.5309$_{(30)}$	0.5540	0.6793$_{(86)}$	0.2730	0.3383$_{(81)}$
$\mathcal{A}_3 \rightarrow \{< NN.* \mid JJ \mid VBG \mid VBN > * < NN.* >\}$	0.5321	0.5330$_{(26)}$	0.5791	0.6770$_{(76)}$	0.2822	0.3288$_{(71)}$
*Pre-trained Keyphrase Predictor for **Truncated** Document (with length limitation→512)*						
$\mathcal{B}_1$ →Pre-trained Keyphrase Generator (T5, *document-level*)	0.2901	0.2901$_{(8)}$	0.3425	0.3538$_{(38)}$	0.3490	0.3756$_{(62)}$
$\mathcal{B}_2$ →Pre-trained Keyphrase Extractor (BERT, *document-level*)	0.4107	0.5328$_{(88)}$	0.3082	0.4844$_{(94)}$	0.3597	0.4340$_{(90)}$
♣ *Ensemble Strategies for **Truncated** Document (with length limitation→512)*						
$\mathcal{A}_1 \cup \mathcal{B}_1$	**0.6263**	0.6314$_{(30)}$	0.5826	0.7933$_{(108)}$	**0.3778**	**0.5020**$_{(125)}$
$\mathcal{A}_1 \cup \mathcal{B}_2$	0.6197	**0.6894**$_{(102)}$	**0.5827**	**0.8211**$_{(165)}$	0.2853	0.4353$_{(155)}$
*Regular Expression for **Original** Document (without length limitation)*						
$\mathcal{A}_1^\dagger \rightarrow \{< NN.* \mid JJ > * < NN.* >\}$	0.5492	0.5503$_{(27)}$	0.5845	0.7960$_{(138)}$	0.2885	0.4789$_{(192)}$
$\mathcal{A}_2^\dagger \rightarrow \{< JJ \mid VBG > * < NN.* > 0, 3\}$	0.5407	0.5452$_{(31)}$	0.5540	0.7932$_{(160)}$	0.2730	0.4893$_{(226)}$
$\mathcal{A}_3^\dagger \rightarrow \{< NN.* \mid JJ \mid VBG \mid VBN > * < NN.* >\}$	0.5452	0.5465$_{(27)}$	0.5791	0.7898$_{(140)}$	0.2822	0.4859$_{(201)}$
*Pre-trained Keyphrase Predictor for **Original** Document (without length limitation)*						
$\mathcal{B}_1^\dagger$ →Pre-trained Keyphrase Generator (T5, *sentence-level*)	0.0041	0.0041$_{(11)}$	0.3605	0.3826$_{(49)}$	0.3449	0.4093$_{(90)}$
$\mathcal{B}_2^\dagger$ →Pre-trained Keyphrase Extractor (BERT, *sentence-level*)	0.2781	0.2784$_{(26)}$	0.2796	0.3935$_{(143)}$	0.2187	0.3744$_{(237)}$
♣ *Ensemble Strategies for **Original** Document (without length limitation)*						
$\mathcal{A}_1^\dagger \cup \mathcal{B}_1^\dagger$	0.5354	0.5471$_{(38)}$	0.5831	0.8556$_{(162)}$	**0.3609**	**0.5785**$_{(238)}$
$\mathcal{A}_1^\dagger \cup \mathcal{B}_2^\dagger$	**0.6078**	**0.6228**$_{(47)}$	**0.5833**	**0.8661**$_{(256)}$	0.2853	0.5601$_{(383)}$

Table 3: Results of the different candidate set generation strategies on three benchmark datasets. The best results are bold, and the second results are underlined. Here, **R@M** compares all the keyphrases extracted by the strategy with the ground-truth keyphrases, which means it considers all phrases in the candidate set. Specifically, for Inspec, $\mathcal{A}_1 \cup \mathcal{B}_2$ = 0.6894$_{(102)}$, where 0.6894 indicates the value of R@M, and 102 indicates the average number of keyphrases in candidate sets.

and SemEval2010 datasets (with long documents) is better than that on the Inspec dataset (with short documents). Compared with the underlined results in Table 2, our model has achieved *10.65%*, *7.44%*, and *11.32%* improvement in *F@5*, *F@10*, and *F@15* on the SemEval2010 dataset.

5.5 Ablation Study

Effect of generating candidate sets with different strategies. The details of the candidate generation strategies and the associated performance are reported in Table 3. For easy description, $\mathcal{A}_*$ denotes $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ and $\mathcal{B}_*$ denotes $\mathcal{B}_1$, $\mathcal{B}_2$. We summarize the detailed analysis as follows:

(1) The ensemble candidate set generation strategy obtains higher recall than using $\mathcal{A}_*$ or $\mathcal{B}_*$.

(2) $\mathcal{A}_*$ obtain more stable and higher recall than $\mathcal{B}_*$ in most cases on three benchmark datasets.

(3) $\mathcal{B}_*$ get higher recall scores on the long document dataset, such as the SemEval2010 dataset.

(4) Intuitively, the longer the document, the more the candidate loss is caused by truncation.

Effect of training with different loss functions. As illustrated in Table 4, our ablation study considers the effect of the reconstruction loss ($\mathcal{L}_\mathcal{C}$+$\mathcal{L}_\mathcal{D}$), the margin-based triplet loss ($\mathcal{L}_\mathcal{T}$), and the orthogonal regularization ($\mathcal{L}_{\mathcal{CC}}$ + $\mathcal{L}_{\mathcal{DD}}$ + $\mathcal{L}_{\mathcal{CD}}$) on the

SetMatch	Acc	F1@5	F1@10	F1@15
$\mathcal{L}_\mathcal{C}$+$\mathcal{L}_\mathcal{D}$	0.12	9.96	17.41	20.76
$\mathcal{L}_\mathcal{T}$+$\mathcal{L}_\mathcal{C}$+$\mathcal{L}_\mathcal{D}$	**0.98**	11.13	16.31	20.49
$\mathcal{L}_\mathcal{T}$+$\mathcal{L}_{\mathcal{CC}}$+$\mathcal{L}_{\mathcal{DD}}$+$\mathcal{L}_{\mathcal{CD}}$	0.81	12.32	18.66	22.93
$\mathcal{L}_\mathcal{T}$+$\mathcal{L}_{\mathcal{CC}}$+$\mathcal{L}_{\mathcal{DD}}$+$\mathcal{L}_{\mathcal{CD}}$+$\mathcal{L}_\mathcal{C}$+$\mathcal{L}_\mathcal{D}$	0.96	**14.44**	**20.79**	**24.18**

Table 4: Performance of training the neural keyphrase set function $\mathcal{F}_s$ by using different loss functions. The best results are in bold.

SemEval2010 dataset. To verify the effectiveness of the neural keyphrase set function directly, we propose a simple method to construct the pseudo label l_i,

$$l_i = \begin{cases} 1 & \textit{if } score(\mathcal{C}_i^+, \mathcal{S}_i^r) > score(\mathcal{C}_i^-, \mathcal{S}_i^r) \\ \text{-1} & \textit{if } score(\mathcal{C}_i^+, \mathcal{S}_i^r) < score(\mathcal{C}_i^-, \mathcal{S}_i^r) \\ 0 & \textit{if } score(\mathcal{C}_i^+, \mathcal{S}_i^r) = score(\mathcal{C}_i^-, \mathcal{S}_i^r) \end{cases} \quad (12)$$

where $\mathcal{S}_i^r$ is the ground-truth keyphrase set of the i-th document $\mathcal{D}_i$. Here, we calculate the $score(\cdot)$ via F1@M, which takes all the phrases in the candidate set $\mathcal{C}$ to evaluate F1 score. After obtaining pseudo labels, we use the keyphrase set function to predict scores following Eq. 12 instead of F1@M, verifying the effectiveness of our keyphrase set function by comparing the predicted scores with pseudo labels for acquiring accuracy. From the

SetMatch	Acc	F1@5	F1@10	F1@15
$Positive: \mathcal{A}_1^{\dagger}, Negative: \mathcal{A}_1$	**0.96**	**14.44**	**20.79**	**24.18**
$Positive: \mathcal{A}_2^{\dagger}, Negative: \mathcal{A}_2$	0.91	14.10	19.69	22.09
$Positive: \mathcal{A}_3^{\dagger}, Negative: \mathcal{A}_3$	0.93	14.32	20.08	22.17

Table 5: Effect of training the keyphrase set function by using different training samples on the SemEval2010 dataset.

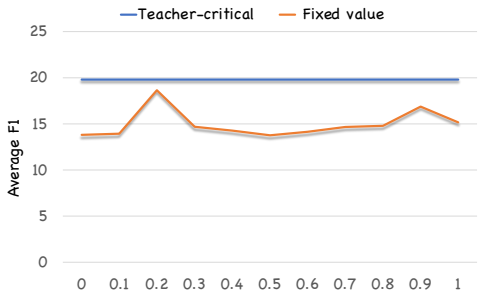

Figure 5: Results of comparing the teacher-critical training strategy with a series of fixed values.

results in Table 4, we can find that $\mathcal{L}_{\mathcal{T}}$ can distinguish positive and negative samples well, and the orthogonal regularization significantly improves the performance.

Effect of different training samples. We adopt different positive and negative samples to train the keyphrase set function $\mathcal{F}_s$, as illustrated in Table 5. The best results are obtained by using $\mathcal{A}_1^{\dagger}$ and $\mathcal{A}_1$.

Effect of the teacher-critical training strategy. To verify the effectiveness of the proposed teacher-critical training strategy, we adopt a series of fixed values to regularize the reward $\mathcal{R}(\mathcal{S}, \mathcal{D})$. Figure 5 shows the results under different values of the regularization on the SemEval2010 dataset. The best results are achieved by using our teacher-critical training strategy. However, dropping the regularization (i.e., the fixed value is set to 0) of the reward $\mathcal{R}(\mathcal{S}, \mathcal{D})$ will significantly damage the final performance. Moreover, our model can be treated as an optimization model for the SOTA UKE baselines by adopting the teacher-critical training strategy.

5.6 Diversity Evaluation

To evaluate the diversity, we follow the previous studies (Bahuleyan and Asri, 2020) and define two evaluation metrics:

(1) **Duplicate**% $= (1 - \frac{\text{\# Unique Tokens}}{\text{\# Extracted Tokens}}) \times 100$

(2) **EditDist**[6]: String matching can be carried out at the character level. Through this evaluation metric, we can calculate the pairwise Levenshtein Distance between extracted keyphrases.

[6] https://github.com/seatgeek/fuzzywuzzy. We utilize the *fuzzywuzzy* library, which calculates a score between 0 and 100, where 100 means exactly matching keyphrases.

Model	Duplicate%@15	EditDist@15
	Inspec	
JointGL(Liang et al., 2021)	34.91	32.77
SetMatch	**28.55**	**32.14**
Ground Truth	14.95	31.37
	DUC2001	
JointGL(Liang et al., 2021)	31.60	34.53
SetMatch	**27.96**	**33.01**
Ground Truth	13.60	31.65
	SemEval2010	
JointGL(Liang et al., 2021)	54.90	44.48
SetMatch	**39.88**	**35.63**
Ground Truth	15.88	30.56

Table 6: Diversity evaluation on three benchmark. The lower value is better in diversity. Note that Porter Stemming is applied before evaluation.

As shown in Table 6, the results demonstrate that globally extracting keyphrases from a set-wise perspective can avoid the repeated selection of phrases with high-frequency words and consider the coupling of multiple keyphrases.

5.7 Case Study

To further provide an intuitive understanding of how our model benefits from a set-wise perspective, we present an example in Table 7. In the given an example, "trajectories" and "feature" are high-frequency words in the document. Therefore, if keyphrases are extracted individually from a point-wise perspective, the phrases containing these two words will get a higher score and be extracted as the keyphrases. However, from a set-wise perspective, it will alleviate the above issue and extract diverse keyphrases. These results further demonstrate that it is effective to extract keyphrases via the document-set matching framework.

6 Related Work

Unsupervised keyphrase extraction approaches can be mainly categorized into statistics-, graph-, and embedding-based methods (Hasan and Ng, 2014; Papagiannopoulou and Tsoumakas, 2019; Song et al., 2023). The statistics-based methods (Jones, 2004; Campos et al., 2018) exploit to use various features (e.g., word frequency, position, linguistic, etc.) for capturing context information. Graph-based methods (Mihalcea and Tarau, 2004; Bougouin et al., 2013; Florescu and Caragea, 2017; Boudin, 2018) usually convert the document into a graph and rank candidate phrases in the graph.

Ground Truth Keyphrase: (1) event detection (2) word trajectory (3) aperiodic event (4) periodic event (5) word signal (6) spectral analysis (7) topic detection (8) topic tracking (9) text streams (10) news stream (11) time series
A point-wise perspective (Liang et al., 2021): (1) feature trajectories (2) ***word trajectories*** (3) ***event detection*** (4) document frequency (5) time-series word signal (6) feature trends (7) periodic features (8) different event characteristics (9) ***periodic events*** (10) retrospective event detection (11) aperiodic words (12) identical trends (13) ***time series*** (14) ***spectral analysis*** (15) representative features
A set-wise perspective (SetMatch): (1) ***event detection*** (2) document frequency (3) time-series word signal (4) unsupervised greedy event detection algorithm (5) ***text streams*** (6) ***aperiodic event*** (7) aperiodic word (8) ***word trajectories*** (9) time series data (10) ***word trajectory*** (11) retrospective event detection (12) ***spectral analysis*** (13) ***topic detection*** (14) unsupervised manner (15) inverse document frequency

Table 7: An example from the SemEval2010 dataset. The bold and italic phrases are the correctly predicted keyphrases by the corresponding models.

Recently, embedding-based methods (Bennani-Smires et al., 2018; Saxena et al., 2020; Sun et al., 2020; Liang et al., 2021; Ding and Luo, 2021; Song et al., 2022b; Zhang et al., 2022), benefiting from the development of pre-trained embeddings (Mikolov et al., 2013; Peters et al., 2018; Devlin et al., 2019), have achieved significant performance. Bennani-Smires et al. (2018) ranks and extracts phrases by estimating the similarities between the embeddings of phrases and the document. Sun et al. (2020) improves embeddings via the pre-trained language model (i.e., ELMo (Peters et al., 2018)) instead of static embeddings (i.e., Word2Vec (Mikolov et al., 2013)). Ding and Luo (2021) models the phrase-document relevance from different granularities via attention weights of the pre-trained language model BERT. Liang et al. (2021) enhances the phrase-document relevance with a boundary-aware phrase centrality to score each phrase in the candidate set individually. Zhang et al. (2022) leverages a masking strategy and ranks candidates by the textual similarity between embeddings of the source document and the masked document. Unlike existing UKE models, we propose to extract keyphrases from a set perspective by learning a neural keyphrase set function, which globally extracts a keyphrase set from the candidate set of the document.

7 Conclusion and Future Work

We formulate the unsupervised keyphrase extraction task as a document-set matching problem and propose a novel set-wise framework to match the document and candidate subsets sampled in the candidate set. It is intractable to exactly search the optimal subset by the document-set matching function, and we thereby propose an approximate algorithm for efficient search which learns a keyphrase set extractor agent via reinforcement learning. Extensive experimental results show SetMatch outperforms the current state-of-the-art unsupervised keyphrase extraction baselines on three benchmark keyphrase extraction datasets, which demonstrates the effectiveness of our proposed paradigm.

Lately, the emergence of Large Language Models (LLMs) has garnered significant attention from the computational linguistics community. For future research, exploring effectively utilizing LLMs to generate candidates and rank candidates to extract keyphrases may be an exciting and valuable direction (i.e., exploring LLM-based UKE).

8 Acknowledgments

We thank the three anonymous reviewers for carefully reading our paper and their insightful comments and suggestions. This work was partly supported by the Fundamental Research Funds for the Central Universities (2019JBZ110); the National Natural Science Foundation of China under Grant 62176020; the National Key Research and Development Program (2020AAA0106800); the Beijing Natural Science Foundation under Grant L211016; CAAI-Huawei MindSpore Open Fund; and Chinese Academy of Sciences (OEIP-O-202004).

9 Limitations

In this paper, we propose a novel set-wise framework to extract keyphrases globally. To verify the effectiveness of the new framework, we design simple yield effective neural networks for both the neural keyphrase set function and the keyphrase set extractor agent modules. In general, a complex neural network should yield better performance. Moreover, for the sake of fairness, our model adopts the same pre-trained language model (i.e., BERT) as the recent state-of-the-art baselines (Liang et al., 2021; Ding and Luo, 2021; Zhang et al., 2022). Actually, other pre-trained language models can be applied to our model, such as RoBERTa (Liu et al., 2019). These pre-trained language models may yield better results, which also demonstrates that there is much room for improvement in our proposed framework. Therefore, we believe the power of this set-wise framework has not been fully exploited. In the future, more forms of document-set matching models can be explored to instantiate the set-wise framework.

References

Hareesh Bahuleyan and Layla El Asri. 2020. Diverse keyphrase generation with neural unlikelihood training. *CoRR*, abs/2010.07665.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *CoNLL*, pages 221–229. Association for Computational Linguistics.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. In *NAACL-HLT (2)*, pages 667–672. Association for Computational Linguistics.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *IJCNLP*, pages 543–551. Asian Federation of Natural Language Processing / ACL.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 343–351, Red Hook, NY, USA. Curran Associates Inc.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *ECIR*, volume 10772 of *Lecture Notes in Computer Science*, pages 806–810. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Haoran Ding and Xiao Luo. 2021. Attentionrank: Unsupervised keyphrase extraction using self and cross attentions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1928.

Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *ACL (1)*, pages 1105–1115. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *ACL (1)*, pages 1262–1273. The Association for Computer Linguistics.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *EMNLP*.

Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *SemEval@ACL*, pages 21–26. The Association for Computer Linguistics.

Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. Unsupervised keyphrase extraction by jointly modeling local and global context. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 155–164, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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. *CoRR*, abs/1907.11692.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *EMNLP*, pages 404–411. ACL.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Eirini Papagiannopoulou and Grigorios Tsoumakas. 2019. A review of keyphrase extraction. *CoRR*, abs/1905.05044.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237. Association for Computational Linguistics.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, pages 1179–1195. IEEE Computer Society.

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

Mingyang Song, Yi Feng, and Liping Jing. 2022a. 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, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5710–5720. Association for Computational Linguistics.

Mingyang Song, Yi Feng, and Liping Jing. 2022b. Utilizing BERT intermediate layers for unsupervised keyphrase extraction. In *Proceedings of the 5th International Conference on Natural Language and Speech Processing (ICNLSP 2022)*, pages 277–281, Trento, Italy. Association for Computational Linguistics.

Mingyang Song, Yi Feng, and Liping Jing. 2023. A survey on recent advances in keyphrase extraction from pre-trained language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2153–2164, Dubrovnik, Croatia. Association for Computational Linguistics.

Mingyang Song, Liping Jing, and Lin Xiao. 2021. Importance Estimation from Multiple Perspectives for Keyphrase Extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Si Sun, Zhenghao Liu, Chenyan Xiong, Zhiyuan Liu, and Jie Bao. 2021. Capturing global informativeness in open domain keyphrase extraction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 275–287. Springer.

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*, pages 855–860. AAAI Press.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256.

Binbin Xie, Xiangpeng Wei, Baosong Yang, Huan Lin, Jun Xie, Xiaoli Wang, Min Zhang, and Jinsong Su. 2022. WR-ONE2SET: towards well-calibrated keyphrase generation. *CoRR*, abs/2211.06862.

Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021. One2set: Generating diverse keyphrases as a set. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4598–4608. Association for Computational Linguistics.

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

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
8

☑ A2. Did you discuss any potential risks of your work?
8

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

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

B ☑ Did you use or create scientific artifacts?

5

☑ B1. Did you cite the creators of artifacts you used?
5

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
5

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
5

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
5

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
5

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
5

C ☑ Did you run computational experiments?

5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
5

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

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
5

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

Left blank.

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

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

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

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

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