An Integrated Approach for Keyphrase Generation via Exploring the Power of Retrieval and Extraction

Wang Chen[1], **Hou Pong Chan**[1], **Piji Li**[2], **Lidong Bing**[3], **Irwin King**[1]
[1]The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2]Tencent AI Lab
[3]R&D Center Singapore, Machine Intelligence Technology, Alibaba DAMO Academy
[1]{wchen, hpchan, king}@cse.cuhk.edu.hk
[2]pijili@tencent.com, [3]l.bing@alibaba-inc.com

Abstract

In this paper, we present a novel integrated approach for keyphrase generation (KG). Unlike previous works which are purely extractive or generative, we first propose a new multi-task learning framework that jointly learns an extractive model and a generative model. Besides extracting keyphrases, the output of the extractive model is also employed to rectify the copy probability distribution of the generative model, such that the generative model can better identify important contents from the given document. Moreover, we retrieve similar documents with the given document from training data and use their associated keyphrases as external knowledge for the generative model to produce more accurate keyphrases. For further exploiting the power of extraction and retrieval, we propose a neural-based merging module to combine and re-rank the predicted keyphrases from the enhanced generative model, the extractive model, and the retrieved keyphrases. Experiments on the five KG benchmarks demonstrate that our integrated approach outperforms the state-of-the-art methods.

1 Introduction

Keyphrases are short text pieces that can quickly express the key ideas of a given document. The keyphrase generation task aims at automatically generating a set of keyphrases given a document. As shown in the upper part of Figure 1, the input is a document and the output is a set of keyphrases. Due to the concise and precise expression, keyphrases are beneficial to extensive downstream applications such as text summarization (Zhang et al., 2004; Wang and Cardie, 2013), sentiment analysis (Wilson et al., 2005; Berend, 2011), and document clustering (Hulth and Megyesi, 2006; Hammouda et al., 2005).

Existing methods on keyphrase generation can be divided into two categories: *extractive* and *generative*. Extractive methods (Medelyan et al., 2009; Mihalcea and Tarau, 2004; Zhang et al., 2016; Luan et al., 2017) identify present keyphrases that appear in the source text like "parameter control" in Figure 1. Although extractive methods are simple to implement, they cannot predict absent keyphrases which are not in the document like "optimization" in Figure 1. Generative methods (Meng et al., 2017; Chen et al., 2018a; Ye and Wang, 2018; Yuan et al., 2018) adopt the well-known encoder-decoder generative model (Luong et al., 2015; Bahdanau et al., 2014) with copy mechanism (Gu et al., 2016; See et al., 2017) to produce keyphrases. In a generative model, the decoder generates keyphrases word by word through either selecting from a predefined vocabulary according to a language model or copying from the source text according to the copy probability distribution computed by a copy mechanism. Thus, these generative methods are capable of generating both present and absent keyphrases.

From a high-level perspective, extractive methods directly locate essential phrases in the docu-

Document:
Futility-Based Offspring Sizing. **Parameter control** in **evolutionary algorithms** (EAs) has been shown to be beneficial; however, the control of offspring size has so far received very little attention. This paper introduces Futility-Based **Offspring Sizing** (FuBOS), a method for controlling offspring size on a per generation basis without even requiring the user to set an initial offspring size value. . .
Keyphrases:
{**evolutionary algorithm**; parameterless evolutionary algorithm; **parameter control**; **offspring sizing**; optimization}

Retrieved Document:
An Exploration into Dynamic **Population Sizing**. Traditional **evolutionary algorithms** are powerful problem solvers that have several fixed parameters which require prior specification. . . While many methods of **parameter control** have been published that focus on removing the population size parameter, μ, all hampered by a variety of problems. This paper investigates the benefits of making μ a dynamic parameter and introduces two novel methods for population control. . .
Retrieved Keyphrases:
{**evolutionary algorithm**; parameterless evolutionary algorithm; **parameter control**; **population sizing**; optimization}

Figure 1: An example of keyphrase generation and retrieval. The present keyphrases are bold.

Proceedings of NAACL-HLT 2019, pages 2846–2856
Minneapolis, Minnesota, June 2 - June 7, 2019.

ment while generative models try to understand the document first and then produce keyphrases. To the best of our knowledge, these two kinds of methods have been developing independently without any combinations among them.

However, when human annotators are asked to assign keyphrases to a document, they usually first obtain a global sense about which parts of the document are important and then write down the keyphrases word by word based on a more detailed understanding. To achieve such a goal, we propose a multi-task learning framework to take advantage of both extractive and generative models. For keyphrase extraction, we adopt a neural sequence labeling model to output the likelihood of each word in the source text to be a keyphrase word (or the importance score of each word). These importance scores are then employed to rectify the copy probability distribution of the generative model. Since the extractive model is explicitly trained to identify keyphrases from the source text, its importance scores can help the copy mechanism to identify important source text words more accurately. Different from the copy probability distribution which is dynamic at each generation step, these importance scores are static. Therefore, they can provide a global sense about which parts of the document are important. In addition, these scores are also utilized to extract present keyphrases which will be exploited by the merging module.

Moreover, human annotators can also incorporate relevant external knowledge like the keyphrases of similar documents that they read before to assign more appropriate keyphrases. Correspondingly, to incorporate external knowledge, we propose a *retriever* to retrieve similar documents of the given document from training data. For instance, as shown in Figure 1, we retrieve a document from the **KP20k** training dataset that has the highest similarity with the upper document. The retrieved document is assigned with almost the same keyphrases as the upper document. Therefore, keyphrases from similar documents (i.e., retrieved keyphrases) can give useful knowledge to guide the generation of keyphrases for the given document. More concretely, we encode the retrieved keyphrases as vector representations and use them as an external memory for the decoder of the generative model in our multi-task learning framework. Besides providing external knowledge, the retrieved keyphrases themselves are regarded as a kind of keyphrase prediction and can be utilized by the merging module.

Finally, to imitate the integrated keyphrase assignment process of humans more comprehensively, we further exploit the extractive model and the retrieved keyphrases by proposing a merging module. This merging module collects and re-ranks the predictions from our aforementioned components. First, keyphrase candidates are collected from three different sources: (1) keyphrases generated by the enhanced generative model; (2) keyphrases extracted by the extractive model; and (3) the retrieved keyphrases. Then, we design a neural-based merging algorithm to merge and re-rank all the keyphrase candidates, and consequently return the top-ranked candidates as our final keyphrases.

We extensively evaluate the performance of our proposed approach on five popular benchmarks. Experimental results demonstrate the effectiveness of the extractive model and the retrieved keyphrases in our multi-task learning framework. Furthermore, after introducing the merging module, our integrated approach consistently outperforms all the baselines and becomes the new state-of-the-art approach for keyphrase generation.

In summary, our main contributions include: (1) a new multi-task learning framework that leverages an extractive model and external knowledge to improve keyphrase generation; (2) a novel neural-based merging module that combines the predicted keyphrases from extractive, generative, and retrieval methods to further improve the performance; and (3) the new state-of-the-art performance on five real-world benchmarks.

2 Related Work

2.1 Automatic Keyphrase Extraction

Keyphrase extraction focuses on predicting the keyphrases that are present in the source text. Existing methods can mainly be categorized into two-step extraction approaches and sequence labeling models. Two-step extraction approaches first identify a set of candidate phrases from the document using different heuristics, such as the phrases that match specific part-of-speech (POS) tags (Liu et al., 2011; Wang et al., 2016; Le et al., 2016). Then, they learn a score for each candidate and select the top-ranked candidates as predicted keyphrases. The scores can be learned by

either supervised methods with hand-crafted textual features (Medelyan et al., 2009; Witten et al., 1999; Nguyen and Kan, 2007; Frank et al., 1999; Hulth, 2003) or unsupervised graph ranking methods (Mihalcea and Tarau, 2004; Grineva et al., 2009; Wan and Xiao, 2008). Sequence labeling models are built on a recurrent neural network to sequentially go through a source text and learn the likelihood of each word in the source text to be a keyphrase word (Zhang et al., 2016; Luan et al., 2017; Gollapalli et al., 2017). In contrast to these extractive methods, our approach can generate both absent and present keyphrases.

2.2 Automatic Keyphrase Generation

Keyphrase generation aims at predicting both present and absent keyphrases for a source text. Meng et al. (2017) proposed CopyRNN, which is built on the attentional encoder-decoder model (Bahdanau et al., 2014) with copy mechanism (Gu et al., 2016) to generate keyphrases. CorrRNN (Chen et al., 2018a), an extension of CopyRNN, was proposed to model the correlations among keyphrases. This model utilizes hidden states and attention vectors of previously generated keyphrases to avoid generating repetitive keyphrases. The title information of the source text was explicitly exploited by Ye and Wang (2018) and Chen et al. (2018b) to further improve the performance. Ye and Wang (2018) first considered a semi-supervised setting for keyphrase generation. In contrast, inspired by Hsu et al. (2018) and Cao et al. (2018), we enhance existing generative methods by adopting an extractive model to assist the copy mechanism and exploiting external knowledge from retrieved keyphrases to help the generation. Furthermore, we also design a merging module to combine the predictions from different components.

3 Our Methodology

As shown in Figure 2, our integrated framework consists of a retriever, two encoders, an extractor, a decoder, and a merging module. Given a document $\mathbf{x}$, the retriever returns the keyphrases $\mathbf{r}$ retrieved from the training corpus. In addition to acting as keyphrase candidates, these retrieved keyphrases are also exploited to provide external guidance for the decoder. Then keyphrase extraction and generation are jointly conducted by the extractor and the decoder through sharing an encoder. Besides extracting keyphrase candidates, the importance scores of the source text words, $\boldsymbol{\beta}$, predicted by the extractor are also employed to rescale the original copy probability distribution of the decoder. Thus, they can help the copy mechanism to detect important words more accurately. Finally, the merging module merges the candidates from three different sources (i.e., the retrieved, extracted, and generated candidates) and output the final predictions.

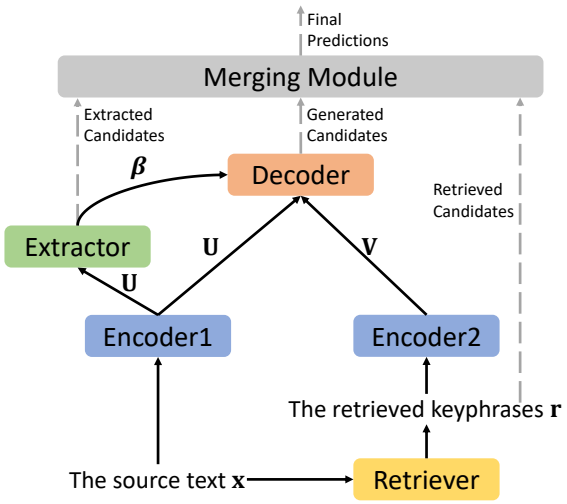

Figure 2: Our integrated framework. The "Encoder1" and the "Extractor" compose our extractive model. Our generative model mainly includes the "Encoder1", the "Encoder2", and the "Decoder".

3.1 Retriever

Given a document $\mathbf{x}$, the retriever module retrieves top K (document, keyphrases) pairs from the training corpus. The retrieval is based on the Jaccard similarities of the non-stop-word sets between $\mathbf{x}$ and the corpus documents. After that, the keyphrases of the top K pairs are returned and used in the later modules in two ways. First, these retrieved keyphrases are regarded as the keyphrase candidates of $\mathbf{x}$ and directly fed into the final merging module. In addition, these keyphrases are concatenated together as a guidance input $\mathbf{r}$ for the decoder to provide useful external knowledge for the generation process. A separator token is inserted among keyphrases when concatenating them together.

3.2 Joint Extraction and Generation

We propose a multi-task learning framework which simultaneously learns to extract keyphrases from the source text and generate keyphrases word

by word with external knowledge. Before describing in detail, we first define the tasks of the extraction and the generation.

3.2.1 Problem Definition

The inputs of the multi-task learning framework are the source text $\mathbf{x}$ and the concatenated retrieved keyphrases $\mathbf{r}$. Both $\mathbf{x}$ and $\mathbf{r}$ are a sequence of tokens (i.e., $\mathbf{x} = [x_1, ..., x_{L_\mathbf{x}}]$, $\mathbf{r} = [r_1, ..., r_{L_\mathbf{r}}]$), where $L_\mathbf{x}$ and $L_\mathbf{r}$ are the length of $\mathbf{x}$ and $\mathbf{r}$ respectively. The output of the extractor is a sequence of importance scores $\boldsymbol{\beta} = [\beta_1, ..., \beta_{L_\mathbf{x}}]$, where β_i is the probability of the i-th source word of being a keyphrase word. The output of the generator is a set of keyphrases $\mathcal{Y} = \{\mathbf{y}^i\}_{i=1,..,N}$, where N is the keyphrase number of $\mathbf{x}$ and $\mathbf{y}^i = [y^i_1, ..., y^i_{L_{\mathbf{y}^i}}]$ is a token sequence with length $L_{\mathbf{y}^i}$.

To fit the encoder-decoder framework, N tuples $\{(\mathbf{x}, \mathbf{r}, \boldsymbol{\beta}^*, (\mathbf{y}^i)^*)\}_{i=1,...,N}$ are split during training, where $\boldsymbol{\beta}^*$ and $(\mathbf{y}^i)^*$ are the gold binary importance scores and one of the gold keyphrases of $\mathbf{x}$ correspondingly. For simplicity, we adopt $(\mathbf{x}, \mathbf{r}, \boldsymbol{\beta}^*, \mathbf{y}^*)$ to represent such a tuple.

3.2.2 Encoders

Two encoders are employed in our multi-task learning framework. One is for the source text encoding (i.e., "Encoder1" in Figure 2) and the other is for retrieved keyphrases encoding (i.e., "Encoder2" in Figure 2). Both of them employ a bidirectional GRU (Cho et al., 2014) layer to obtain a context-aware representation of each word:

$$\mathbf{u}_i = \text{BiGRU}_1(\mathbf{x}_i, \overrightarrow{\mathbf{u}}_{i-1}, \overleftarrow{\mathbf{u}}_{i+1}), \quad (1)$$

$$\mathbf{v}_j = \text{BiGRU}_2(\mathbf{r}_j, \overrightarrow{\mathbf{v}}_{j-1}, \overleftarrow{\mathbf{v}}_{j+1}), \quad (2)$$

where $i = 1, 2, ..., L_\mathbf{x}$ and $j = 1, 2, ..., L_\mathbf{r}$. $\mathbf{x}_i$ and $\mathbf{r}_j$ are the d_e-dimensional embedding vectors of the i-th source text word x_i and j-th retrieved keyphrases word r_j respectively. $\mathbf{u}_i = [\overrightarrow{\mathbf{u}}_i; \overleftarrow{\mathbf{u}}_i] \in \mathbb{R}^d$ and $\mathbf{v}_j = [\overrightarrow{\mathbf{v}}_j; \overleftarrow{\mathbf{v}}_j] \in \mathbb{R}^d$ are regarded as the corresponding context-aware representations, where d is the hidden size of the biderectional GRU layer. Finally, we obtain the internal memory bank $\mathbf{U} = [\mathbf{u}_1, ..., \mathbf{u}_{L_\mathbf{x}}]$ for later extraction and generation, and the external memory bank $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_{L_\mathbf{r}}]$ for later generation.

3.2.3 Extractor

Based on the internal memory bank, we use the following sequence identifier as our extractor to identify whether the word is a keyphrase word in the source text. We denote the importance score $P(\beta_j = 1|\mathbf{u}_j, \mathbf{s}_j, \mathbf{d})$ as β_j for simplicity:

$$\begin{aligned}\beta_j = \text{sigmoid}(&\mathbf{W}_c\mathbf{u}_j + \mathbf{u}_j^T\mathbf{W}_s\mathbf{d} \\ &- \mathbf{u}_j^T\mathbf{W}_n\text{tanh}(\mathbf{s}_j) + b),\end{aligned} \quad (3)$$

where $\mathbf{d} = \tanh(\mathbf{W}_d[\overrightarrow{\mathbf{u}}_{L_\mathbf{x}}; \overleftarrow{\mathbf{u}}_1] + \mathbf{b})$ is the global document representation and $\mathbf{s}_j = \sum_{i=1}^{j-1} \mathbf{u}_i\beta_i$ is current summary representation. $\mathbf{W}_c$, $\mathbf{W}_s$, and $\mathbf{W}_n$ are the content, salience and novelty weights respectively. Although this extractor is inspired by Nallapati et al. (2017), our extractor identifies important words instead of sentences within the source text.

3.2.4 Decoder

In addition to the internal memory bank $[\mathbf{u}_1, ..., \mathbf{u}_{L_\mathbf{x}}]$, our decoder employs the external memory bank $[\mathbf{v}_1, ..., \mathbf{v}_{L_\mathbf{r}}]$ to provide external guidance for the generation process. We exploit a decoder equipped with attention and copy mechanisms (Luong et al., 2015; See et al., 2017) to generate keyphrases. This decoder mainly consists of a forward GRU layer:

$$\mathbf{h}_t = \overrightarrow{\text{GRU}}([\mathbf{e}_{t-1}; \tilde{\mathbf{h}}_{t-1}], \mathbf{h}_{t-1}), \quad (4)$$

$$\mathbf{c}_t^{in} = \text{attn}(\mathbf{h}_t, [\mathbf{u}_1, ..., \mathbf{u}_{L_\mathbf{x}}], \mathbf{W}_{in}), \quad (5)$$

$$\mathbf{c}_t^{ex} = \text{attn}(\mathbf{h}_t, [\mathbf{v}_1, ..., \mathbf{v}_{L_\mathbf{r}}], \mathbf{W}_{ex}), \quad (6)$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_1[\mathbf{c}_t^{in}; \mathbf{c}_t^{ex}]; \mathbf{h}_t), \quad (7)$$

where $\mathbf{e}_{t-1}$ is the embedding vector of the $(t-1)$-th predicted word. The "attn" operation in Eq. (5) is defined as $\mathbf{c}_t^{in} = \sum_{i=1}^{L_\mathbf{x}} \alpha_{t,i}^{in}\mathbf{u}_i$, where $\alpha_{t,i}^{in} = \exp(s_{t,i}) / \sum_{j=1}^{L_\mathbf{x}} \exp(s_{t,j})$ and $s_{t,i} = (\mathbf{h}_t)^T\mathbf{W}_{in}\mathbf{u}_i$. Similarly, we can obtain the external aggregated vector $\mathbf{c}_t^{ex}$.

Then, the final predicted probability distribution at the current time step is:

$$P(y_t) = (1 - g_t)P_v(y_t) + g_tP_c(y_t), \quad (8)$$

where $g_t = \sigma(\mathbf{w}_g^T\tilde{\mathbf{h}}_t + b_g) \in \mathbb{R}$ is the soft switch between generating from the predefined vocabulary V and copying from X that are all words appearing in the source text. $P_v(y_t) = \text{softmax}(\mathbf{W}_2\tilde{\mathbf{h}}_t + \mathbf{b}_v) \in \mathbb{R}^{|V|}$ is the generating probability distribution over V and $P_c(y_t) = \sum_{i:x_i=y_t} \alpha_{t,i}^c \in \mathbb{R}^{|X|}$ is the copying probability distribution over X. Previous work either directly uses the internal attention scores as the copy probabilities (i.e., $\alpha_{t,i}^c = \alpha_{t,i}^{in}$) or employs extra neural network layers to calculate new copy scores. But

we employ the rescaled internal attention scores α_t^{in} by the importance scores $[\beta_1, ..., \beta_{L_\mathbf{x}}]$ from the extractor as the final copy probabilities:

$$\alpha_{t,i}^{c} = \frac{\alpha_{t,i}^{in} * \beta_i}{\sum_{j=1}^{L_\mathbf{x}} \alpha_{t,j}^{in} * \beta_j}. \quad (9)$$

The purpose of this rescaling is to provide extra guidance that which words within the source text are important and thus should obtain more attention when copying.

3.2.5 Joint Training

Finally, the summation of the following extraction loss and generation loss is used to train the whole joint framework in an end-to-end way.

Extraction Loss. We choose the source text words appearing in the assigned keyphrases as the gold important words and use the weighted cross-entropy loss for the extraction training i.e., $\mathcal{L}_e = -\frac{1}{L_\mathbf{x}} \sum_{j=1}^{L_\mathbf{x}} w\beta_j^* \log\beta_j + (1-\beta_j^*)\log(1-\beta_j)$, where $\beta_j^* \in \{0, 1\}$ is the ground-truth label for the j-th word and w is the loss weight for the positive training samples.

Generation Loss. The negative log likelihood loss is utilized for the generation training i.e., $\mathcal{L}_g = -\sum_{t=1}^{L_{\mathbf{y}^*}} \log P(y_t^* | \mathbf{y}_{t-1}, \mathbf{x}, \mathbf{r})$, where $\mathbf{y}_t = [y_1, ..., y_{t-1}]$ is the previously predicted word sequence, $L_{\mathbf{y}^*}$ is the length of target keyphrase $\mathbf{y}^*$, and y_t^* is the t-th target word in $\mathbf{y}^*$.

3.3 Merging Module

In this module, the retrieved, extracted and generated keyphrases are collected and then merged to produce the final keyphrase predictions.

3.3.1 Keyphrase Candidate Collection

Retrieved Candidate Collection. The retrieved keyphrases from the retriever are regarded as the retrieved candidates. Each retrieved candidate (rk) obtains a retrieval score (rs) that is the Jaccard similarity between the corresponding document and $\mathbf{x}$. The duplicates with lower retrieval scores are removed. Finally, we get $N_\mathbf{rk}$ retrieved keyphrase candidates $\mathbf{rk} = [rk_1, \ldots, rk_{N_\mathbf{rk}}]$ and their retrieval scores $\mathbf{rs} = [rs_1, \ldots, rs_{N_\mathbf{rk}}]$.

Extracted Candidate Collection. The extracted keyphrase candidates are from the extractor. We select the word x_j as a keyword if its importance score β_j is larger or equal than a threshold ϵ (i.e., $\beta_j \geq \epsilon$). The adjacent keywords compound a keyphrase candidate. If no other adjacent keywords, the keyword itself becomes a single-word keyphrase candidate. Each extracted keyphrase candidate (ek) is accompanied by an extraction score (es) that is the mean of the importance scores of the words within this candidate. Similarly, duplicates with lower extraction scores are removed. Consequently, we obtain $N_\mathbf{ek}$ extracted keyphrase candidates $\mathbf{ek} = [ek_1, \ldots, ek_{N_\mathbf{ek}}]$ and the corresponding extraction scores $\mathbf{es} = [es_1, \ldots, es_{N_\mathbf{ek}}]$.

Generated Candidate Collection. The generated keyphrase candidates directly come from the beam search process of the decoder. Each generated phrase is a keyphrase candidate. The beam search score of the generated candidate (gk) represents its generation score (gs). Duplicates with lower generation scores are removed. Then, we get $N_\mathbf{gk}$ generated candidates $\mathbf{gk} = [gk_1, \ldots, gk_{N_\mathbf{gk}}]$ and their generation scores $\mathbf{gs} = [gs_1, \ldots, gs_{N_\mathbf{gk}}]$.

3.3.2 Merging

In addition to the original importance scores (i.e., $\mathbf{rs}, \mathbf{es}, \mathbf{gs}$), we also employ an auxiliary *scorer* to assign an auxiliary importance score to each keyphrase candidate. Given a document-candidate pair ($\mathbf{x}$, candidate), the scorer should output the probability that the candidate is one of the keyphrases of $\mathbf{x}$. That means the scorer should determine the relationship between the given document $\mathbf{x}$ and the candidate, which is similar to a natural language inference (NLI) problem. Therefore, we adapt the most popular NLI model (Parikh et al., 2016) as our scorer. Different from typical natural language inference which is a multi-class classification problem, we use a binary classification setting to train the scorer. Besides, we learn the word embeddings and use two bi-directional GRU to obtain the input representations. The positive samples are the ground-truth keyphrases. The negative samples come from either the phrases in the document or the retrieved candidates. Notably, the ground-truth keyphrases are filtered when selecting negative samples. Consequently, a cross-entropy loss is utilized to train the scorer. Finally, the trained scorer is used to help the merging process as shown in Algorithm 1. The $\frac{u_\mathbf{gs}}{u_\mathbf{rs}}$ and $\frac{u_\mathbf{gs}}{u_\mathbf{es}}$ factors are used to enforce the average of $\mathbf{rs}$ and $\mathbf{es}$ to be the same with the average of $\mathbf{gs}$ and thus these three scores become comparable.

Algorithm 1 Merging Algorithm

Require: The retrieved, extracted and generated candidates $\mathbf{rk}, \mathbf{ek}, \mathbf{gk}$. The retrieval, extraction and generation scores $\mathbf{rs}, \mathbf{es}, \mathbf{gs}$; The average of each kind of score: $u_{\mathbf{rs}}, u_{\mathbf{es}}, u_{\mathbf{gs}}$; The trained $scorer$; The document $\mathbf{x}$.

1: Adjust $\mathbf{gs}$: $gs_i = gs_i \times scorer(\mathbf{x}, gk_i)$ where $i = 1, \ldots, N_{\mathbf{gk}}$.
2: Adjust $\mathbf{rs}$: $rs_i = rs_i \times \frac{u_{\mathbf{gs}}}{u_{\mathbf{rs}}} \times scorer(\mathbf{x}, rk_i)$ where $i = 1, \ldots, N_{\mathbf{rk}}$.
3: Adjust $\mathbf{es}$: $es_i = es_i \times \frac{u_{\mathbf{gs}}}{u_{\mathbf{es}}} \times scorer(\mathbf{x}, ek_i)$ where $i = 1, \ldots, N_{\mathbf{ek}}$.
4: Merge $\mathbf{rk}, \mathbf{ek}, \mathbf{gk}$: the final importance score of a candidate is the summation of its adjusted retrieval, extraction and generation scores. If not in $\mathbf{rk}$, $\mathbf{ek}$ or $\mathbf{gk}$, the corresponding scores are set to 0.
5: Sort all the candidates based on the final importance scores and then output the final predictions.

4 Experiment Settings

4.1 Datasets

Similar to Meng et al. (2017), we use **KP20k** dataset (Meng et al., 2017) to train our models. The released dataset contains 530,809 articles for training, 20,000 for validation, and the other 20,000 for testing. However, there exist duplicates in the **KP20k** training dataset with itself, the **KP20k** validation dataset, the **KP20k** testing dataset, and other four popular testing datasets (i.e., **Inspec** (Hulth, 2003), **Krapivin** (Krapivin et al., 2009), **NUS** (Nguyen and Kan, 2007), and **SemEval** (Kim et al., 2010)). After removing these duplicates, we maintain 509,818 articles in the training dataset. As for testing, following Meng et al. (2017), we employ five popular testing datasets from scientific publications as our testbeds for the baselines and our methods, which include **Inspec**, **Krapivin**, **NUS**, **SemEval**, and **KP20k**.

4.2 Baseline Models and Evaluation Metrics

For a comprehensive evaluation, we compare our methods with the traditional extractive baselines and the state-of-the-art generative methods. The extractive baselines include two unsupervised methods (i.e., TF-IDF and TextRank (Mihalcea and Tarau, 2004)) and one supervised method Maui (Medelyan et al., 2009). The generative baselines consist of CopyRNN (Meng et al., 2017) and CorrRNN (Chen et al., 2018a). We also conduct several ablation studies as follows:

- **KG-KE.** The joint extraction and generation model without using the retrieved keyphrases and merging process.
- **KG-KR.** The encoder-decoder generative model with retrieved keyphrases as external knowledge, but without combining with the extractive model and using the merging process.
- **KG-KE-KR.** The joint extraction and generation model with the retrieved keyphrases without using the merging process.

All the above ablation models directly use the generated candidates as the final predictions. We denote our final integrated method which combines all the proposed modules as **KG-KE-KR-M**.

Similar to CopyRNN and CorrRNN, we adopt macro-averaged *recall* (R) and *F-measure* (F_1) as our evaluation metrics. In addition, we also apply Porter Stemmer before determining whether two keyphrases are identical. Duplications are removed after stemming.

4.3 Implementation Details

We apply similar preprocessing procedures with Meng et al. (2017) including lowercasing, tokenizing and replacing digits with $\langle digit \rangle$ symbol. The title and the abstract of each article are concatenated as the source text input. We use the **KP20k** training dataset as the retrieval corpus. The implementations of our models are based on the OpenNMT system (Klein et al., 2017). The encoders, the decoder, and the scorer have the same vocabulary V with 50,000 tokens. The multi-task learning model and the scorer are trained separately.

The embedding dimension d_e and the hidden size d are set to 100 and 300 respectively. The initial state of the decoder GRU cell (i.e., $\mathbf{h}_0$) is set to $[\overrightarrow{\mathbf{u}}_{L_\mathbf{x}}; \overleftarrow{\mathbf{u}}_1]$. The other GRU cells are set to zero. The retrieval number K is set to 3 after evaluating the retrieved keyphrases on the evaluation dataset. When concatenating the retrieved keyphrases together as an external knowledge input, we use ';' as the separator among them. During training, all the trainable parameters including the embeddings are randomly initialized with uniform distribution in [-0.1, 0.1]. We engage Adam (Kingma and Ba, 2014) as the optimizer with positive extraction loss weight w=9.0, batch size=64, dropout rate=0.1, max gradient norm=1.0, initial learning rate=0.001. The training is early stopped when the validation perplexity stops dropping for several continuous evaluations. While testing, the beam search depth, and beam size are set as 6 and 200

Model	Inspec		Krapivin		NUS		SemEval		KP20k	
	$F_1@5$	$F_1@10$	$F_1@5$	$F_1@10$	$F_1@5$	$F_1@10$	$F_1@5$	$F_1@10$	$F_1@5$	$F_1@10$
TF-IDF	0.188	0.269	0.092	0.120	0.103	0.142	0.076	0.135	0.087	0.113
TextRank	0.194	0.244	0.142	0.128	0.147	0.153	0.107	0.130	0.151	0.132
Maui	0.037	0.032	0.196	0.181	0.205	0.234	0.032	0.036	0.223	0.204
CorrRNN*	0.229_{7}	0.248_{9}	0.255_{2}	0.238_{4}	0.273_{5}	0.265_{4}	0.197_{3}	0.221_{5}	0.291_{2}	0.264_{2}
CopyRNN*	0.251_{7}	0.279_{3}	0.268_{4}	0.243_{1}	0.275_{2}	0.268_{2}	0.190_{6}	0.214_{5}	0.306_{1}	0.273_{0}
KG-KE	0.254_{4}	0.281_{2}	0.265_{3}	0.240_{1}	0.278_{4}	0.273_{1}	$\mathbf{0.207_{4}}$	$\mathbf{0.227_{7}}$	0.307_{0}	0.274_{0}
KG-KR	0.244_{2}	0.275_{1}	0.266_{5}	0.247_{1}	0.278_{2}	0.276_{2}	0.189_{7}	0.215_{7}	0.311_{1}	0.278_{0}
KG-KE-KR	0.245_{1}	0.278_{4}	0.267_{3}	0.246_{2}	0.285_{9}	0.279_{4}	0.194_{4}	0.220_{2}	0.314_{0}	0.280_{0}
KG-KE-KR-M	$\mathbf{0.257_{2}}$	$\mathbf{0.284_{3}}$	$\mathbf{0.272_{3}}$	$\mathbf{0.250_{2}}$	$\mathbf{0.289_{4}}$	$\mathbf{0.286_{4}}$	0.202_{6}	0.223_{3}	$\mathbf{0.317_{0}}$	$\mathbf{0.282_{0}}$

Table 1: Total keyphrase prediction results on all testing datasets. The best results are bold and the second best results are underlined. The subscripts are corresponding standard deviations for neural-based models (e.g. 0.257_2 means 0.257 ± 0.002). The '*' indicates our implementations based on Luong et al. (2015) attention and See et al. (2017) copying. The implementations of our proposed models are based on "CopyRNN*".

Model	Inspec	Krapivin	NUS	SemEval	KP20k
TF-IDF	0.141	0.069	0.069	0.043	0.064
TextRank	0.158	0.110	0.094	0.062	0.110
Maui	0.024	0.162	0.161	0.012	0.196
CorrRNN*	0.172_{6}	0.217_{6}	0.212_{4}	0.125_{3}	0.268_{3}
CopyRNN*	0.195_{7}	0.229_{3}	0.216_{8}	0.120_{8}	0.285_{1}
KG-KE	0.197_{3}	0.225_{3}	0.219_{3}	$\mathbf{0.135_{5}}$	0.287_{0}
KG-KR	0.190_{1}	0.228_{5}	0.222_{7}	0.120_{6}	0.293_{0}
KG-KE-KR	0.191_{3}	0.229_{3}	0.224_{5}	0.127_{5}	0.295_{1}
KG-KE-KR-M	$\mathbf{0.201_{2}}$	$\mathbf{0.234_{2}}$	$\mathbf{0.234_{6}}$	0.131_{4}	$\mathbf{0.299_{0}}$

Table 2: MAP@10 scores of total keyphrase predictions. The best results are bold and the second best results are underlined. The meanings of the subscripts and the '*' are the same as in Table 1.

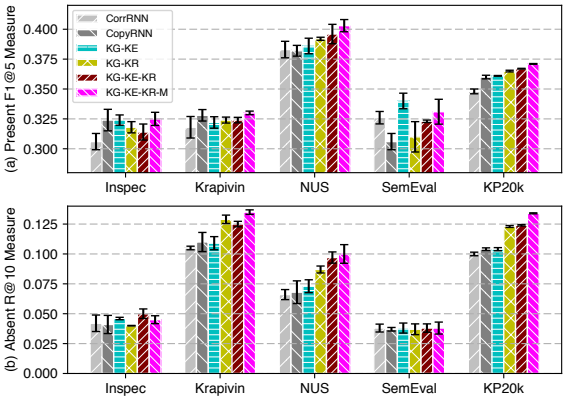

Figure 3: The present and absent keyphrase prediction performance of all neural-based methods.

correspondingly. The extraction threshold ϵ is set to 0.7 after evaluating the extracted keyphrases on the evaluation dataset. Notably, the stemmer is not applied to the gold keyphrases of **SemEval** testing dataset since they have already been stemmed. We do not remove any single-word predictions for **KP20k** but only keep one single-word prediction for other testing datasets. The averaged results of three different random seed are reported[1].

5 Results and Analysis

5.1 Total Keyphrase Prediction

Unlike the previous works which only separately analyze the present and absent keyphrase prediction ability, we also compare the whole keyphrase prediction ability regardless of the presence or absence of keyphrases, which is more reasonable in real applications. We show the F_1 scores at the top 5 and 10 predictions on Table 1.

This table displays our KG-KE-KR-M method consistently outperforms the state-of-the-art models CopyRNN and CorrRNN demonstrating the effectiveness of our method. Moreover, we also observe that the KG-KE model exceeds CopyRNN and CorrRNN on most datasets, which indicates the strength of our combination with the extractive model. Besides, we also see the KG-KR model perform comparably or better than the baselines, suggesting the effective guidance ability of the retrieved keyphrases. In addition, after combining these two ideas, the KG-KE-KR model surpasses both or one of KG-KE and KG-KR on all datasets, which shows the effectiveness of the combination with extraction model and the retrieved keyphrases again. Finally, the performance gap between KG-KE-KR and KG-KE-KR-M implies the power of our merging module. For mean average precision (MAP) metric which considers prediction orders, we obtain similar conclusions as shown in Table 2.

5.2 Present and Absent Keyphrase Prediction

In this section, we analyze the performance of present and absent keyphrase prediction. Only

[1]Our code is available at https://github.com/Chen-Wang-CUHK/KG-KE-KR-M

Candidate Sources	Total F_1@10	Present F_1@5	Absent R@10
gk, ek, rk	**0.250±0.002**	**0.330±0.002**	**0.172±0.002**
gk, ek	0.249±0.003	0.328±0.003	0.154±0.002
gk, rk	0.249±0.002	0.329±0.002	0.172±0.002
gk	0.248±0.003	0.327±0.003	0.154±0.002
gk, no merging	0.246±0.002	0.324±0.002	0.158±0.002
ek, no merging	0.152±0.005	0.226±0.010	N/A
rk, no merging	0.093±0.000	0.121±0.000	0.107±0.000

Table 3: Ablation study of the candidate sources of Algorithm 1 on Krapivin dataset. "no merging" means we do not use the merging algorithm.

Scoring Method	Total F_1@10	Present F_1@5	Absent R@10
Combined	**0.250±0.002**	**0.330±0.002**	**0.172±0.002**
Only **gs, es, rs**	0.248±0.003	0.325±0.003	0.166±0.003
Only *scorer*	0.210±0.005	0.291±0.006	0.106±0.005

Table 4: Ablation study of the scoring method of Algorithm 1 on Krapivin dataset. "Only **gs, es, rs**" means we do not use the *scorer*. "Only *scorer*" represents we directly use the scores predicted by the *scorer* as the final importance scores.

the present (absent) predictions and gold present (absent) keyphrases are preserved for the corresponding evaluation. We use F_1@5 metric for present predictions and R@10 for absent predictions. Since the neural-based baselines are the state-of-the-art models, we focus on the comparison with them in this section. The results are depicted on Figure 3.

The main observations are similar to the conclusions of total keyphrase prediction. Besides, we also note that after incorporating retrieved keyphrases, KG-KR model achieves substantial improvement gains over baselines on absent keyphrase prediction on **Krapivin**, **NUS**, and **KP20k**. These results demonstrate that the retrieved keyphrases indeed help the model to understand the main topics of the given document since generating absent keyphrase is an abstractive process and requires more powerful text understanding abilities. We notice that the KG-KE-KR-M method does not outperform the KG-KE-KR model on absent keyphrase prediction on **Inspec** dataset. One potential reason is that the merging module only merges two sources for absent keyphrases (i.e., the generated and retrieved keyphrases) instead of three sources like the present keyphrases do. Hence, the improvement for the absent keyphrases from the merging module is less stable than that for the present keyphrases. Moreover, we find that after combining with the extraction model, the KG-KE model achieves a huge improvement gain over CopyRNN on present keyphrase prediction on **SemEval** dataset, which manifests such a combination can improve the keyphrase extraction ability of the generative model.

5.3 Ablation Study on Merging Module

We also conduct in-depth ablation studies on our merging module. The objectives of these ablation studies are to (1) evaluate the effects of different candidate sources (i.e., what kinds of candidates are merged), and (2) analyze the effects of different final importance score calculating methods.

Concerning candidate sources, we show the ablation study results on Table 3. When comparing "**gk**" with "**gk**, no merging", we can see that the merging algorithm improves the performance of total and present keyphrase predictions, but it degrades the performance of absent keyphrase prediction. These results indicate the trained *scorer* performs better on scoring present keyphrases than scoring absent keyphrases. One possible reason is that scoring absent keyphrases requires a stronger text understanding ability than scoring present keyphrases. However, as shown in the row of "**gk, rk**" on Table3, this problem can be solved by incorporating the retrieved keyphrases which provide external information to this module. Besides absent keyphrase prediction, it is observed that the retrieved keyphrases can also benefit the present keyphrase prediction. For the extracted keyphrases, as shown in the "**gk, ek**" row, they only improve the present keyphrase prediction ability and do not affect absent keyphrases as we anticipated.

Regarding the scoring method, we further explore the effects of not using or only using the *scorer* in Algorithm 1. We show the results on Table 4. From this table, we note that after removing the *scorer* (i.e., "Only **gs, es, rs**"), both present and absent keyphrase prediction performance become worse, which demonstrates the effectiveness of the combination with the *scorer*. Moreover, if we totally ignore the previously obtained retrieval, extraction and generation scores, and only use the *scorer* to predict the final keyphrase importance score (i.e., "Only *scorer*"), we find the performance decreases dramatically, which indicates the indispensability of the previously obtained retrieval, extraction, and generation scores.

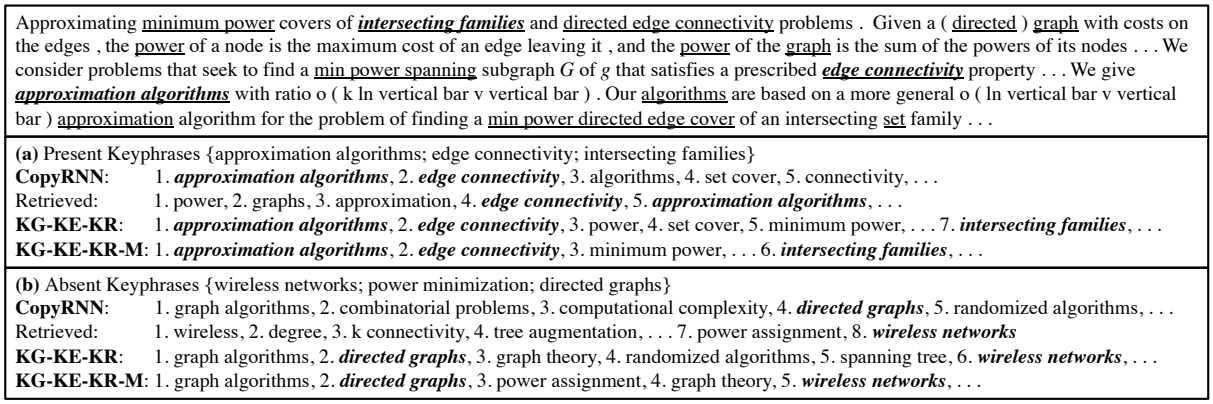

Figure 4: A keyphrase prediction example of CopyRNN, KG-KE-KR, and KG-KE-KR-M. "Retrieved" is the retrieved keyphrases. The extracted keyphrases by the extractor of KG-KE-KR are underlined in the source text. Top 10 present and absent predictions are compared and some incorrect predictions are omitted for simplicity. The correct predictions are bold and italic.

5.4 Case Study

To illustrate the advantages of our proposed methods, we show an example of the present and absent keyphrase predictions in Figure 4. For fairness, we only compare with CopyRNN since our models are based on its implementation. From the results of the present keyphrase prediction, we find the extractor of the KG-KE-KR model successfully extracts all the present keyphrases from the source text, which shows the power of the extractor. With the help of the copy probability rescaling from the extractor, the KG-KE-KR model correctly predicts the keyphrase "intersecting families" which is not successfully predicted by CopyRNN and retrieved by the retriever. Moreover, by merging the extracted keyphrases into the final predictions, the KG-KE-KR-M model assigns a higher rank to this keyphrase (i.e., from 7 to 6). As for absent keyphrase prediction, we note that KG-KE-KR successfully predicts the keyphrase "wireless networks" while CopyRNN fails. Since the retriever successfully retrieves this absent keyphrase, it shows that the retrieved keyphrases can provide effective external guidance for the generation process. Furthermore, the KG-KE-KR-M method assigns a higher rank to this keyphrase after merging the retrieved keyphrases into the final predictions (i.e., from 6 to 5). The overall results demonstrate the effectiveness of our proposed methods.

6 Conclusion and Future Work

In this paper, we propose a novel integrated approach for keyphrase generation. First, an end-to-end multi-task learning framework is introduced, which not only combines the keyphrase extraction and generation but also leverages the retrieved keyphrases from similar documents to guide the generation process. Furthermore, we introduce a neural-based merging algorithm to merge the candidates from three different components. Comprehensive empirical studies demonstrate the effectiveness of our approach. One interesting future work is to incorporate the similar documents themselves into keyphrase generation.

Acknowledgments

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 of the General Research Fund) and Meitu (No. 7010445). We would like to thank Jiani Zhang for her comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*, pages 1162–1170.

Ziqiang Cao, Wenjie Li, Furu Wei, and Sujian Li. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages

152–161. Association for Computational Linguistics.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018a. Keyphrase generation with correlation constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4057–4066.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. 2018b. Title-guided encoding for keyphrase generation. *CoRR*, abs/1808.08575.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 668–673.

Sujatha Das Gollapalli, Xiaoli Li, and Peng Yang. 2017. Incorporating expert knowledge into keyphrase extraction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3180–3187.

Maria P. Grineva, Maxim N. Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 661–670.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Khaled M. Hammouda, Diego N. Matute, and Mohamed S. Kamel. 2005. Corephrase: Keyphrase extraction for document clustering. In *Machine Learning and Data Mining in Pattern Recognition, 4th International Conference, MLDM 2005, Leipzig, Germany, July 9-11, 2005, Proceedings*, pages 265–274.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141. Association for Computational Linguistics.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2003, Sapporo, Japan, July 11-12, 2003*.

Anette Hulth and Beáta Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

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, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, pages 21–26.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, System Demonstrations*, pages 67–72.

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction. Technical report, University of Trento.

Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. 2016. Unsupervised keyphrase extraction: Introducing new kinds of words to keyphrases. In *AI 2016: Advances in Artificial Intelligence - 29th Australasian Joint Conference, Hobart, TAS, Australia, December 5-8, 2016, Proceedings*, pages 665–671.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning, CoNLL 2011, Portland, Oregon, USA, June 23-24, 2011*, pages 135–144.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017. Scientific information extraction with semi-supervised neural tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2641–2651.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1318–1327.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 582–592.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 404–411.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3075–3081.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers, 10th International Conference on Asian Digital Libraries, ICADL 2007, Hanoi, Vietnam, December 10-13, 2007, Proceedings*, pages 317–326.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 855–860.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1395–1405.

Minmei Wang, Bo Zhao, and Yihua Huang. 2016. PTR: phrase-based topical ranking for automatic keyphrase extraction in scientific publications. In *Neural Information Processing - 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16-21, 2016, Proceedings, Part IV*, pages 120–128.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 347–354.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM conference on Digital Libraries, August 11-14, 1999, Berkeley, CA, USA*, pages 254–255.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4142–4153.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Daqing He, and Adam Trischler. 2018. Generating diverse numbers of diverse keyphrases. *CoRR*, abs/1810.05241.

Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 836–845.

Yongzheng Zhang, A. Nur Zincir-Heywood, and Evangelos E. Milios. 2004. World wide web site summarization. *Web Intelligence and Agent Systems*, 2(1):39–53.