Heterogeneous Graph Neural Networks for Keyphrase Generation

Jiacheng Ye[1]*, **Ruijian Cai**[1*], **Tao Gui**[2†] **and Qi Zhang**[1†]
[1]School of Computer Science, Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai, China
[2]Institute of Modern Languages and Linguistics, Fudan University
{yejc19, 19210240253, tgui, qz}@fudan.edu.cn

Abstract

The encoder-decoder framework achieves state-of-the-art results in keyphrase generation (KG) tasks by predicting both present keyphrases that appear in the source document and absent keyphrases that do not. However, relying solely on the source document can result in generating uncontrollable and inaccurate absent keyphrases. To address these problems, we propose a novel graph-based method that can capture explicit knowledge from related references. Our model first retrieves some document-keyphrases pairs similar to the source document from a pre-defined index as references. Then a heterogeneous graph is constructed to capture relationships of different granularities between the source document and its references. To guide the decoding process, a hierarchical attention and copy mechanism is introduced, which directly copies appropriate words from both the source document and its references based on their relevance and significance. The experimental results on multiple KG benchmarks show that the proposed model achieves significant improvements against other baseline models, especially with regard to the absent keyphrase prediction.

1 Introduction

Keyphrase generation (KG), a fundamental task in the field of natural language processing (NLP), refers to the generation of a set of keyphrases that expresses the crucial semantic meaning of a document. These keyphrases can be further categorized into present keyphrases that appear in the document and absent keyphrases that do not. Current KG approaches generally adopt an encoder-decoder framework (Sutskever et al., 2014) with attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) and copy mechanism (Gu et al., 2016; See et al., 2017) to simultaneously predict present and absent keyphrases (Meng et al., 2017; Chen et al., 2018; Chan et al., 2019; Chen et al., 2019b,a; Yuan et al., 2020).

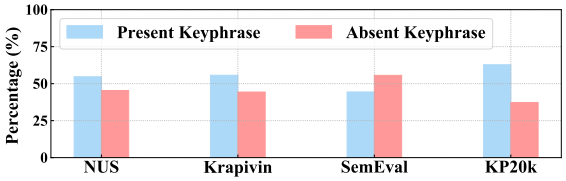

Figure 1: Proportion of present and absent keyphrases among four datasets. Although the previous methods for keyphrase generation have shown promising results on present keyphrase predictions, they are not yet satisfactory on the absent keyphrase predictions, which also occupy a large proportion.

Although the proposed methods for keyphrase generation have shown promising results on present keyphrase predictions, they often generate uncontrollable and inaccurate predictions on the absent ones. The main reason is that there are numerous candidates of absent keyphrases that have implicit relationships (e.g., technology hypernyms or task hypernyms) with the concepts in the document. For instance, for a document discussing "LSTM", all the technology hypernyms like "Neural Network", "RNN" and "Recurrent Neural Network" can be its absent keyphrases candidates. When dealing with scarce training data or limited model size, it is non-trivial for the model to summarize and memorize all the candidates accurately. Thus, one can expect that the generated absent keyphrases are often sub-optimal when the candidate set in model's mind is relatively small or inaccurate. This problem is crucial because absent keyphrases account for a large proportion of all the ground-truth keyphrases. As shown in Figure 1, in some datasets, up to 50% of the keyphrases are absent.

To address this problem, we propose a novel graph-based method to capture explicit knowl-

* Equal contribution.
† Corresponding authors.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 2705–2715
November 7–11, 2021.

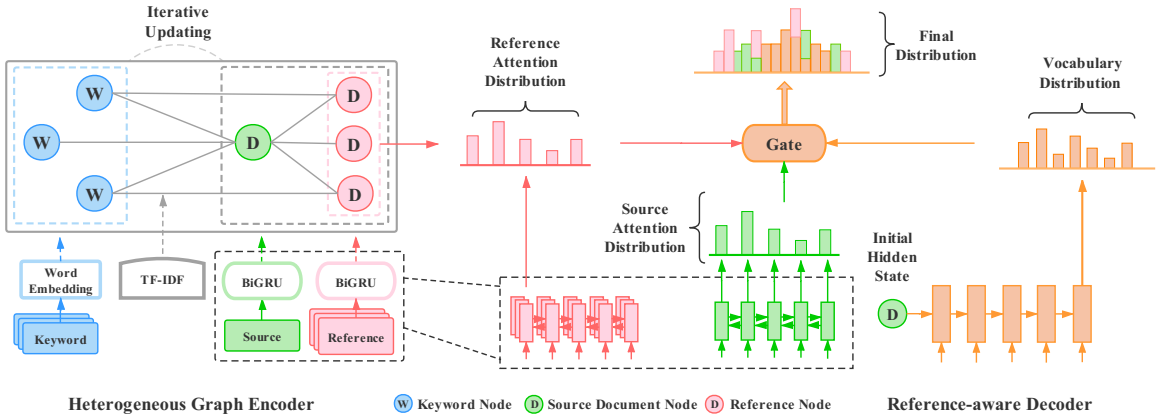

Figure 2: Graphical illustration of our proposed GATER. We first retrieve references using the source document, where each reference is the concatenation of document and keyphrases pair from the training set. Then we construct a heterogeneous graph and perform iterative updating. Finally, the source document node is extracted to decode the keyphrase sequence with a hierarchical attention and copy mechanism.

edge from related references. Each reference is a retrieved document-keyphrases pair from a predefined index (e.g., the training set) that similar to the source document. This is motivated by the fact that the related references often contain candidate or even ground-truth absent keyphrases of the source document. Empirically, we find three retrieved references cover up to 27% of the ground-truth absent keyphrases on average (see Section 4.3 for details).

Our heterogeneous graph is designed to incorporate knowledge from the related references. It contains source document, reference and keyword nodes, and has the following advantages: (a) different reference nodes can interact with the source document regarding the explicit shared keyword information, which can enrich the semantic representation of the source document; (b) a powerful structural prior is introduced as the keywords are highly overlapped with the ground-truth keyphrases. Statistically, we collect the top five keywords from each document on the validation set, and we find that these keywords contain 68% of the tokens in the ground-truth keyphrases. On the decoder side, as a portion of absent keyphrases directly appear in the references, we propose a hierarchical attention and copy mechanism for copying appropriate words from both source document and its references based on their relevance and significance.

The main contributions of this paper can be summarized as follows: (1) we design a heterogeneous graph network for keyphrase generation, which can enrich the source document node through keyword nodes and retrieved reference nodes; (2) we propose a hierarchical attention and copy mechanism to facilitate the decoding process, which can copy appropriate words from both the source document and retrieved references; and (3) our proposed method outperforms other state-of-the-art methods on multiple benchmarks, and especially excels in absent keyphrase prediction. Our codes are publicly available at *Github*[1].

2 Methodology

In this work, we propose a heterogeneous Graph ATtention network basEd on References (GATER) for keyphrase generation, as shown in Figure 2. Given a source document, we first retrieve related document from a predefined index[2] and concatenate each retrieved document with its keyphrases to serve as a reference. Then we construct a heterogeneous graph that contains document nodes[3] and keyword nodes based on the source document and its references. The graph is updated iteratively to enhance the representations of the source document node. Finally, the source document node is extracted to decode the keyphrase sequence. To facilitate the decoding process, we also introduce a hierarchical attention and copy mechanism, with which the model directly attends to and copies from both the source document and its references. The hierarchical ar-

[1] https://github.com/jiacheng-ye/kg_gater

[2] We use the training set as our reference index in our experiment, which can also be easily extended to open corpus.

[3] Note that source document and reference are the two specific contents of the document node.

rangement ensures that more semantically relevant words and those in more relevant references will be given larger weights for the current decision.

2.1 Reference Retriever

Given a source document $\mathbf{x}$, we first use a reference retriever to output several related references from the training set. To make full use of both the retrieved document and retrieved keyphrases, we denote a reference as the concatenation of the two. We find that the use of a term frequency–inverse document frequency (TF-IDF)-based retriever provides a simple but efficient means to accomplish the retrieval task. Specifically, we first represent the source document and all the reference candidates as TF-IDF weighted uni/bi-gram vectors. Then, the most similar K references $\mathcal{X}^r = \{\mathbf{x}^{r_i}\}_{i=1,\dots,K}$ are retrieved by comparing the cosine similarities of the vectors of the source document and all the references.

2.2 Heterogeneous Graph Encoder

2.2.1 Graph Construction

Given the source document $\mathbf{x}$ and its references $\mathcal{X}^r$, we select the top-k unique words as keywords based on their TF-IDF weights from the source document and each reference. The additional keyword nodes can enrich the semantic representation of the source document through message passing, and introduce prior knowledge for generating keyphrase as the highly overlap between keywords and keyphrases. We then build a heterogeneous graph based on the source document, references and keywords.

Formally, our undirected heterogeneous graph can be defined as $G = \{V, E\}$, $V = V_w \cup V_d$ and $E = E_{d2d} \cup E_{w2d}$. Specifically, $V_w = \{w_i\}$ ($i \in \{1, \dots, m\}$) denotes m unique keyword nodes of the source document and K references, $V_d = \mathbf{x} \cup \mathcal{X}^r$ corresponds to the source document node and K reference nodes, $E_{d2d} = \{e_k\}$ ($k \in \{1, \dots, K\}$) and e_k represents the edge weight between the k-th reference and source document, and $E_{w2d} = \{e_{i,j}\}$ ($i \in \{1, \dots, m\}, j \in \{1, \dots, K+1\}$) and $e_{i,j}$ indicates the edge weight between the i-th keyword and the j-th document.

2.2.2 Graph Initializers

Node Initializers There are two types of nodes in our heterogeneous graph (i.e., document nodes V_d and keyword nodes V_w). For each document node, the same as previous works (Meng et al., 2017; Chen et al., 2019a), an embedding lookup table $\mathbf{e}^w$ is first applied to each word, and then a bidirectional Gated Recurrent Unit (GRU) (Cho et al., 2014) is used to obtain the context-aware representation of each word. The representation for document $\mathbf{x}$ and each word is defined as the concatenation of the forward and backward hidden states (i.e., $\mathbf{d} = [\overrightarrow{\mathbf{m}}_1; \overleftarrow{\mathbf{m}}_{L_\mathbf{x}}]$ and $\mathbf{m}_i = [\overrightarrow{\mathbf{m}}_i; \overleftarrow{\mathbf{m}}_i]$, respectively). For each keyword node, since the same keyword may appear in multiple documents, we simply use the word embedding as its initial node representation $\mathbf{w}_i = \mathbf{e}^w(w_i)$.

Edge Initializers There are two types of edges in our heterogeneous graph (i.e., document-to-document edge E_{d2d} and document-to-keyword E_{d2w}). To include information about the significance of the relationships between keyword and document nodes, we infuse TF-IDF values in the edge weights. Similarly, we also infuse TF-IDF values in the edge weights of E_{d2d} as a prior statistical n-gram similarity between documents. The two types of floating TF-IDF weights are then transformed into integers and mapped to dense vectors using embedding matrices $\mathbf{e}^{d2d}$ and $\mathbf{e}^{w2d}$.

2.2.3 Graph Aggregating and Updating

Aggregator Graph attention networks (GAT) (Velickovic et al., 2018) are used to aggregate information for each node. We denote the hidden states of input nodes as $\mathbf{h}_i \in \mathbb{R}^{d_h}$, where $i \in \{1, \dots, N\}$. With the additional edge feature, the aggregator is defined as follows:

$$
\begin{aligned}
z_{ij} &= \text{LeakyReLU}\left(\mathbf{w}_a^T \left[\mathbf{W}_q \mathbf{h}_i; \mathbf{W}_k \mathbf{h}_j; \mathbf{e}_{ij}\right]\right) \\
\alpha_{ij} &= \text{softmax}_j\left(z_{ij}\right) = \frac{\exp\left(z_{ij}\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(z_{ik}\right)} \\
\mathbf{u}_i &= \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}_v \mathbf{h}_j\Big),
\end{aligned}
\tag{1}
$$

where $\mathbf{e}_{ij}$ is the embedding of edge feature, α_{ij} is the attention weight between $\mathbf{h_i}$ and $\mathbf{h_j}$, and $\mathbf{u}_i$ is the aggregated feature. For simplicity, we will use $\text{GAT}(\mathbf{H}, \mathbf{H}, \mathbf{H}, \mathbf{E})$ to denote the GAT aggregating layer, where $\mathbf{H}$ is used for query, key, and value, and $\mathbf{E}$ is used as edge features.

Updater To update the node state, similar to the approach used in the Transformer (Vaswani et al., 2017), we introduce a residual connection and position-wise feed-forward (FFN) layer consisting of two linear transformations. Given an undirected

heterogeneous graph G with node features $\mathbf{H}_w \cup \mathbf{H}_d$ and edge features $\mathbf{E}_{w2d} \cup \mathbf{E}_{d2d}$, we update each types of nodes separately as follows:

$$\begin{aligned}\mathbf{H}_w^1 &= \text{FFN}\left(\text{GAT}\left(\mathbf{H}_w^0, \mathbf{H}_d^0, \mathbf{H}_d^0, \mathbf{E}_{w2d}\right) + \mathbf{H}_w^0\right)\\ \mathbf{H}_d^1 &= \text{FFN}\left(\text{GAT}\left(\mathbf{H}_d^0, \mathbf{H}_w^1, \mathbf{H}_w^1, \mathbf{E}_{w2d}\right) + \mathbf{H}_d^0\right)\\ \mathbf{H}_d^1 &= \text{FFN}\left(\text{GAT}\left(\mathbf{H}_d^1, \mathbf{H}_d^1, \mathbf{H}_d^1, \mathbf{E}_{d2d}\right) + \mathbf{H}_d^1\right),\end{aligned} \tag{2}$$

with word nodes updated first by aggregating document-level information from document nodes, then document nodes updated by the updated word nodes, and finally document nodes updated again by the updated document nodes. The above process is executed iteratively for I steps to realize better document representation.

When the heterogeneous graph encoder finished, we seperate $\mathbf{H}_d^I$ into $\mathbf{d}^s$ and $\mathbf{D}^r = \{\mathbf{d}^{r_i}\}_{i=1...,K}$ as the representation of source document and each reference. We denote $\mathbf{M}^s = \{\mathbf{m}_i^s\}_{i=1,...,L_\mathbf{x}}$ as the encoder hidden state of each word in the source document, $\mathbf{M}^r = \{\mathbf{M}^{r_i}\}_{i=1...,K}$ and $\mathbf{M}^{r_i} = \{\mathbf{m}_j^{r_i}\}_{j=1...,L_{r_i}}$ denotes the encoder hidden state of each word of the i-th reference. All the features described above (i.e., $\mathbf{d}^s$, $\mathbf{D}^r$, $\mathbf{M}^s$ and $\mathbf{M}^r$) will be used in the reference-aware decoder.

2.3 Reference-aware Decoder

After encoding the document into a reference-aware representation $\mathbf{d}^s$, we propose a hierarchical attention and copy mechanism to further incorporate the reference information by attending to and copying words from both the source document and the references.

We use $\mathbf{d}^s$ as the initial hidden state of a GRU decoder, and the decoding process in time step t is described as follows:

$$\begin{aligned}\mathbf{h}_t &= \text{GRU}(\mathbf{e}^w(y_{t-1}), \mathbf{h}_{t-1})\\ \mathbf{c}_t &= \text{hier\_attn}(\mathbf{h}_t, \mathbf{M}^s, \mathbf{M}^r, \mathbf{D}^r)\\ \tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t]),\end{aligned} \tag{3}$$

where $\mathbf{c}_t$ is the context vector and the hierarchical attention mechanism hier_attn is defined as follows:

$$\begin{aligned}&\mathbf{c}_t^s = \sum_{i=1}^{L_\mathbf{x}} a_{t,i}^s \mathbf{m}_i^s; \mathbf{c}_t^r = \sum_{i=1}^{K}\sum_{j=1}^{L_{\mathbf{x}^{r_i}}} a_{t,i}^r a_{t,j}^{r_i} \mathbf{m}_j^{r_i}\\ &\mathbf{c}_t = g_{ref} \cdot \mathbf{c}_t^s + (1 - g_{ref}) \cdot \mathbf{c}_t^r,\end{aligned} \tag{4}$$

where $\mathbf{a}_t^s$ is a word-level attention distribution over words from the source document using $\mathbf{M}^s$, $\mathbf{a}_t^r$ is an attention distribution over references using $\mathbf{D}^r$, which gives greater weights to more relevant references, $\mathbf{a}_t^{r_i}$ is a word-level attention distribution over words from i-th reference using $\mathbf{M}^{r_i}$, which can be considered as the importance of each word in i-th reference, and $g_{ref} = \text{sigmoid}(\mathbf{w}_{ref}[\mathbf{c}_t^s; \mathbf{c}_t^r])$ is a soft gate for determining the importance of the context vectors from source document and references. All the attention distributions described above are computed as in Bahdanau et al. (2015).

To alleviate the out-of-vocabulary (OOV) problem, a copy mechanism (See et al., 2017) is generally adopted. To further guide the decoding process by copying appropriate words from references based on their relevance and significance, we propose a hierarchical copy mechanism. Specifically, a dynamic vocabulary $\mathcal{V}'$ is constructed by merging the predefined vocabulary $\mathcal{V}$, the words in source document $\mathcal{V}_\mathbf{x}$ and all the words in the references $\mathcal{V}_{\mathcal{X}^r}$. Thus, the probability of predicting a word $y_t$is computed as follows:

$$P_{\mathcal{V}'}(y_t) = p_1 P_\mathcal{V}(y_t) + p_2 P_{\mathcal{V}_\mathbf{x}}(y_t) + p_3 P_{\mathcal{V}_{\mathcal{X}^r}}(y_t), \tag{5}$$

where $P_\mathcal{V}(y_t) = \text{softmax}(\text{MLP}([\mathbf{h}_t; \tilde{\mathbf{h}}_t]))$ is the generative probability over predefined vocabulary $\mathcal{V}$, $P_{\mathcal{V}_\mathbf{x}}(y_t) = \sum_{i:x_i=y_t} a_{t,i}^s$ is the copy probability from the source document, $P_{\mathcal{V}_{\mathcal{X}^r}}(y_t) = \sum_i \sum_{j:x_j^{r_i}=y_t} a_{t,j}^{r_i}$ is the copy probability from all the references, and $\mathbf{p} = \text{softmax}(\mathbf{W}_p[\tilde{\mathbf{h}}_t; \mathbf{h}_t; \mathbf{e}^w(y_{t-1})]) \in \mathbb{R}^3$ serves as a soft switcher that determines the preference for selecting the word from the predefined vocabulary, source document or references.

2.4 Training

The proposed GATER model is independent of any specific training method, so we can use either the ONE2ONE training paradigm (Meng et al., 2017), where the target keyphrase set $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1,...,|\mathcal{Y}|}$ are split into multiple training targets for a source document $\mathbf{x}$:

$$\mathcal{L}_{\text{ONE2ONE}}(\theta) = -\sum_{i=1}^{|\mathcal{Y}|}\sum_{t=1}^{L_{\mathbf{y}_i}} \log P_{\mathcal{V}'}(y_{i,t} \mid \mathbf{y}_{i,1:t-1}, \mathbf{x}; \theta), \tag{6}$$

or the ONE2SEQ training paradigm (Ye and Wang, 2018; Yuan et al., 2020), where all the keyphrases

are concatenated into one training target:

$$\mathcal{L}_{\text{ONE2SEQ}}(\theta) = -\sum_{t=1}^{L_{\mathbf{y}^\star}} \log P_{\mathcal{V}'}\left(y_t^\star \mid \mathbf{y}^\star{}_{1:t-1}, \mathbf{x}; \theta\right), \tag{7}$$

where $\mathbf{y}^\star$ is the concatenation of the keyphrases in $\mathcal{Y}$ by a delimiter.

3 Experimental Setup

3.1 Datasets

We conduct our experiments on four scientific article datasets, including **NUS** (Nguyen and Kan, 2007), **Krapivin** (Krapivin et al., 2009), **SemEval** (Kim et al., 2010) and **KP20k** (Meng et al., 2017). Each sample from these datasets consists of a title, an abstract, and some keyphrases given by the authors of the papers. Following previous works (Meng et al., 2017; Chen et al., 2019b,a; Yuan et al., 2020), we concatenate the title and abstract as a source document. We use the largest dataset (i.e., **KP20k**) for model training, and the testing sets of all the four datasets for evaluation. After preprocessing (i.e., lowercasing, replacing all the digits with the symbol $\langle digit \rangle$ and removing the duplicated data), the final **KP20k** dataset contains 509,818 samples for training, 20,000 for validation and 20,000 for testing. The number of test samples in **NUS**, **Krapivin** and **SemEval** is 211, 400 and 100, respectively.

3.2 Baselines

For a comprehensive evaluation, we verify our method under both training paradigms (i.e., ONE2ONE and ONE2SEQ) and compare with the following methods[4]:

- **catSeq** (Yuan et al., 2020). The RNN-based seq2seq model with copy mechanism under ONE2SEQ training paradigm. **CopyRNN** (Meng et al., 2017) is the one with the same model but under ONE2ONE training paradigm.
- **catSeqD** (Yuan et al., 2020). An extension of catSeq with orthogonal regularization (Bousmalis et al., 2016) and target encoding to improve diversity under ONE2SEQ training paradigm.
- **catSeqCorr** (Chan et al., 2019). The extension of catSeq with coverage and review mechanisms under ONE2SEQ training paradigm. **CorrRNN** (Chen et al., 2018) is the one under ONE2ONE training paradigm.
- **catSeqTG** (Chan et al., 2019). The extension of catSeq with additional title encoding. **TG-Net** (Chen et al., 2019b) is the one under ONE2ONE training paradigm.
- **KG-KE-KR-M** (Chen et al., 2019a). A joint extraction and generation model with the retrieved keyphrases and a merging process under ONE2ONE training paradigm.
- **SenSeNet** (Luo et al., 2020). The extension of catSeq with document structure under ONE2SEQ paradigm.

[4]We didn't compare with Chen et al. (2020) since they use a different preprocessing method with others, see the discussion on github for details.

3.3 Implementation Details

Following previous works (Chan et al., 2019; Yuan et al., 2020), when training under the ONE2SEQ paradigm, the target keyphrase sequence is the concatenation of present and absent keyphrases, with the present keyphrases are sorted according to the orders of their first occurrences in the document and the absent keyphrase kept in their original order.

We keep all the parameters the same as those reported in Chan et al. (2019), hence, we only report the parameters in the additional graph module. We retrieve 3 references and extract the top 20 keywords from source document and each reference to construct the graph. We set the number of attention heads to 5 and the number of iterations to 2, based on the valid set. During training, we use a dropout rate of 0.3 for the graph layer, the batch size of 12 and 64 for ONE2SEQ and ONE2ONE training paradigm, respectively. During testing, we use greedy search for ONE2SEQ, and beam search with a maximum depth of 6 and a beam size of 200 for ONE2ONE. We repeat the experiments of our model three times using different random seeds and report the averaged results.

3.4 Evaluation Metrics

For the model trained under ONE2ONE paradigm, as in previous works (Meng et al., 2017; Chen et al., 2018, 2019b), we use macro-averaged $F_1@5$ and $F_1@10$ for present keyphrase predictions, and $R@10$ and $R@50$ for absent keyphrase predictions. For the model trained under ONE2SEQ paradigm, we follow Chan et al. (2019) and use $F_1@5$ and $F_1@M$ for both present and absent keyphrase predictions, where $F_1@M$ compares all the keyphrases predicted by the model with

Model	NUS				SemEval				KP20k			
	Present		Absent		Present		Absent		Present		Absent	
	$F1@5$	$F1@10$	$R@10$	$R@50$	$F1@5$	$F1@10$	$R@10$	$R@50$	$F1@5$	$F1@10$	$R@10$	$R@50$
CopyRNN (Meng et al., 2017)	0.311	0.266	0.058	0.116	0.293	0.304	0.043	0.067	0.333	0.262	0.125	0.211
CorrRNN (Chen et al., 2018)	0.318	0.278	0.059	-	0.320	0.320	0.041	-	-	-	-	-
TG-Net (Chen et al., 2019b)	0.349	0.295	0.075	0.137	0.318	0.322	0.045	0.076	0.372	0.315	0.156	0.268
KG-KE-KR-M (Chen et al., 2019a)	0.344	0.287	0.123	**0.193**	0.329	0.327	0.049	0.090	0.400	**0.327**	0.177	0.278
CopyRNN-GATER (Ours)	$\mathbf{0.374}_4$	$\mathbf{0.304}_4$	$\mathbf{0.126}_3$	$\mathbf{0.193}_2$	$\mathbf{0.366}_3$	$\mathbf{0.340}_4$	$\mathbf{0.056}_1$	$\mathbf{0.092}_2$	$\mathbf{0.402}_1$	0.324_1	$\mathbf{0.186}_0$	$\mathbf{0.285}_1$

Table 1: Keyphrase prediction results of all the models trained under ONE2ONE paradigm. The best results are bold. The subscript are corresponding standard deviation (e.g., 0.285_1 means 0.285 ± 0.001).

Model	NUS				SemEval				KP20k			
	Present		Absent		Present		Absent		Present		Absent	
	$F1@5$	$F1@M$	$F1@5$	$F1@M$	$F1@5$	$F1@M$	$F1@5$	$F1@M$	$F1@5$	$F1@M$	$F1@5$	$F1@M$
catSeq (Yuan et al., 2020)	0.323	0.397	0.016	0.028	0.242	0.283	0.020	0.028	0.291	0.367	0.015	0.032
catSeqD (Yuan et al., 2020)	0.321	0.394	0.014	0.024	0.233	0.274	0.016	0.024	0.285	0.363	0.015	0.031
catSeqCorr (Chan et al., 2019)	0.319	0.390	0.014	0.024	0.246	0.290	0.018	0.026	0.289	0.365	0.015	0.032
catSeqTG (Chan et al., 2019)	0.325	0.393	0.011	0.018	0.246	0.290	0.019	0.027	0.292	0.366	0.015	0.032
SenSeNet (Luo et al., 2020)	**0.348**	0.403	0.018	0.032	0.255	0.299	0.024	0.032	**0.296**	0.370	0.017	0.036
catSeq-GATER (Ours)	0.337_4	$\mathbf{0.418}_4$	$\mathbf{0.033}_3$	$\mathbf{0.054}_4$	$\mathbf{0.257}_3$	$\mathbf{0.309}_4$	$\mathbf{0.026}_4$	$\mathbf{0.035}_5$	0.295_2	$\mathbf{0.384}_1$	$\mathbf{0.030}_1$	$\mathbf{0.060}_2$

Table 2: Keyphrase prediction results of all the models trained under ONE2SEQ paradigm. The best results are bold. The subscript are corresponding standard deviation (e.g., 0.060_2 means 0.060 ± 0.002).

the ground-truth keyphrases, which means it considers the number of predictions. We apply the Porter Stemmer before determining whether two keyphrases are identical and remove all the duplicated keyphrases after stemming.

4 Results and Analysis

4.1 Present and Absent Keyphrase Predictions

Table 1 and Table 2 show the performance evaluations of the present and absent keyphrase predicted by the model trained under ONE2ONE paradigm and ONE2SEQ paradigm, respectively.[5] For the results on absent keyphrases, as noted by previous works (Chan et al., 2019; Yuan et al., 2020) that predicting absent keyphrases for a document is an extremely challenging task, the proposed GATER model still outperforms the state-of-the-art baseline models on all the metrics under both training paradigms, which demonstrates the effectiveness of our methods that includes the knowledge of references. Compared to KG-KE-KR-M, CopyRNN-GATER achieves the same or better results on all the datasets. This suggests that both the retrieved document and keyphrases are useful for predicting absent keyphrases.

For present keyphrase prediction, we find that GATER outperforms most of the baseline methods on both training paradigms, which indicates that the related references also help the model to understand the source document and to predict more accurate present keyphrases.

4.2 Ablation Study

To examine the contribution of each component in GATER, we conduct ablation experiments on the largest dataset **KP20k**, the results of which are presented in Table 3. For the input references, the model's performance is degraded if either the retrieved documents or retrieved keyphrases are removed, which indicates that both are useful for keyphrases prediction. For the heterogeneous graph encoder, the graph becomes a heterogeneous bipartite graph when the $d2d$ edges are removed, and a homogeneous graph when the $w2d$ edges are removed. We can see that both result in degraded performance due to the lack of interaction. Removing both the $d2d$ edges and the $w2d$ edges means that the reference information is only used on the decoder side with the reference-aware decoder, which further degrades the results. For the reference-aware decoder, we find the hierarchical attention and copy mechanism to be essential to the performance of GATER. This indicates the importance of integrating knowledge from references on the decoder side.

4.3 Quality and Influence of References

As our graph is based on the retrieved references, we also investigated the quality and influence of the references. We define the quality of the retrieved references as the *transforming rate* of

[5] Due to the space limitations, the results on the Krapivin dataset can be found in Appendix A.

Model	Present		Absent	
	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$
catSeq-GATER	**0.295**	**0.384**	**0.030**	**0.060**
Input Reference				
- retrieved documents	0.293	0.377	0.026	0.052
- retrieved keyphrases	0.291	0.369	0.018	0.037
- both	0.291	0.367	0.015	0.032
Heterogeneous Graph Encoder				
- $d2d$ edge	0.294	0.379	0.024	0.049
- $w2d$ edge	0.294	0.379	0.026	0.052
- both	0.293	0.371	0.020	0.041
Reference-aware Decoder				
- hierarchical copy	0.293	0.373	0.022	0.042
- hierarchical attention	0.291	0.368	0.018	0.036

Table 3: Ablation study of catSeq-GATER on **KP20k** dataset. All references are ignored in graph encoder when removing $d2d$ edge and the heterogeneous graph becomes homogeneous graph when removing $w2d$ edge.

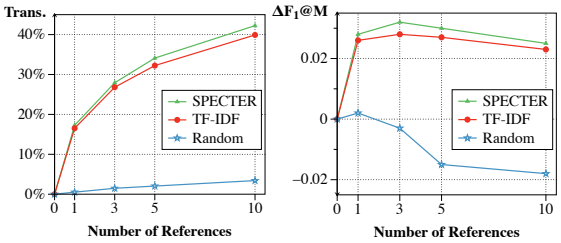

Figure 3: Transforming rate and $\Delta F_1@M$ for absent keyphrases under different types of retrievers on **KP20k** dataset for catSeq-GATER. We study a random retriever, a sparse retriever based on TF-IDF and a dense retriever based on SPECTER.

absent keyphrase (i.e., the proportion of absent keyphrases that appear in the retrieved references). Intuitively, the references that contain more absent keyphrases provide more explicit knowledge for the model generation. As shown on the left part in Figure 3, the simple sparse retriever based on TF-IDF outperforms the random retriever by a large margin regarding the reference quality. We also use a dense retriever SPECTER[6] (Cohan et al., 2020), which is a BERT-based model pretrained using scientific documents. We find that using a dense retriever further helps in the transforming rate of absent keyphrases. On the right part of Figure 3, we show the influence of the references, and we note that random references degrade the model performance as they contain a lot of noise. Surprisingly, we can obtain a 2.6% performance boost in the prediction of absent keyphrase by considering only the most similar references with a sparse or dense retriever, and the introduction of more than three references does not further improve the performance. One possible explanation is that although more references lead to a higher transforming rate of the absent keyphrase, they also introduce more irrelevant information, which interferes with the judgment of the model.

[6]https://github.com/allenai/specter

Model	Present		Absent	
	$F1@5$	$F1@M$	$F1@5$	$F1@M$
catSeqD	0.285	0.363	0.015	0.031
+ GATER	**0.294**	**0.381**	**0.025**	**0.051**
catSeqCorr	0.289	0.365	0.015	0.032
+ GATER	**0.296**	**0.384**	**0.030**	**0.060**
catSeqTG	0.292	0.366	0.015	0.032
+ GATER	**0.293**	**0.380**	**0.025**	**0.052**

Table 4: Results of applying our GATER to other baseline models on **KP20k** test set. The best results are bold.

4.4 Incorporating Baselines with GATER

Our proposed GATER can be considered as an extra plugin for incorporating knowledge from references on both the encoder and decoder sides, which can also be easily applied to other models. We investigate the effects of adding GATER to other baseline models in Table 4. We note that GATER enhances the performance of all the baseline models in both predicting present and absent keyphrases. This further demonstrates the effectiveness and portability of the proposed method.

4.5 Case Study

We display a prediction example by baseline models and CopyRNN-GATER in Figure 4. Our model generates more accurate present and absent keyphrases comparing to the baselines. For instance, we observe that CopyRNN-GATER successfully predicts the absent keyphrase "porous medium" as it appears in the retrieved documents, while both CopyRNN and KG-KE-KR-M fail. This demonstrates that using both the retrieved documents and keyphrases as references provides more knowledge (e.g., candidates of the ground-truth absent keyphrases) compared with using keyphrases alone as in KG-KE-KR-M.

5 Related Work

5.1 Keyphrase Extraction and Generation

Existing approaches for keyphrase prediction can be broadly divided into extraction and generation methods. Early work mostly use a two-step

Document: **natural convection** in porous annular domains **mimetic scheme** and **family of steady states**. **natural convection** of the incompressible fluid in the porous media based on the darcy hypothesis (lapwood convection) gives an intriguing branching off of one parameter family of steady patterns. this scenario may be suppressed in computations when governing equations are approximated by schemes which do not preserve the **cosymmetry** property ...	
Present Keyphrases	natural convection; mimetic scheme; family of steady states; cosymmetry
CopyRNN	**natural convection**; porous media; polar coordinates; mimetic; annular porous domain; porous domain; finite difference; steady states; **mimetic scheme**; ...
KG-KE-KR-M	**natural convection**; porous media; **mimetic scheme**; mimetic; polar coordinates; ...
CopyRNN-G$_{\text{ATER}}$ (Ours)	**natural convection**; porous media; **mimetic scheme**; **cosymmetry**; mimetic; darcy hypothesis; finite difference; polar coordinates; ...
Absent Keyphrases	darcy law; porous medium; finite difference method
CopyRNN	convective convection; annular porous media; mimetic method; **finite difference method**; ...
KG-KE-KR-M	cosymmetry convection; mimetic method; **darcy law**; convective patterns; lapwood property; annular porous media; **finite difference method**; ...
CopyRNN-G$_{\text{ATER}}$ (Ours)	**darcy law**; convective patterns; **porous medium**; multicomponent fluid; **finite difference method**; family convection; cosymmetry convection; staggered grids; darcy formulation; ...

Figure 4: Example of generated keyphrases by different models. The top 10 predictions are compared and some incorrect predictions are omitted for simplicity. The correct predictions are in bold blue and bold red for present and absent keyphrase, respectively. The absent predictions that appear in the references are highlighted in yellow, where only the keyphrases of retrieved documents are considered as references for KG-KE-KR-M.

approach for keyphrase extraction. First, they extract a large set of candidate phrases by hand-crafted rules (Mihalcea and Tarau, 2004; Medelyan et al., 2009; Liu et al., 2011). Then, these candidates are scored and reranked based on unsupervised methods (Mihalcea and Tarau, 2004; Wan and Xiao, 2008) or supervised methods (Hulth, 2003; Nguyen and Kan, 2007). Other extractive approaches utilize neural-based sequence labeling methods (Zhang et al., 2016; Gollapalli et al., 2017).

Keyphrase generation is an extension of keyphrase extraction which considers the absent keyphrase prediction. Meng et al. (2017) proposed a generative model CopyRNN based on the encoder-decoder framework (Sutskever et al., 2014). They employed an ONE2ONE paradigm that uses a single keyphrase as the target sequence. Since CopyRNN uses beam search to perform independently prediction, it's lack of dependency on the generated keyphrases, which results in many duplicated keyphrases. CorrRNN (Chen et al., 2018) proposed a review mechanism to consider the hidden states of the previously generated keyphrase. Ye and Wang (2018) proposed to use a seperator $\langle sep \rangle$ to concatnate all keyphrases as a sequence in training. With this setup, the seq2seq model is capable to generate all possible keyphrases in one sequence as well as capture the contextual information between the keyphrases. However, it still use beam search to generate multiple keyphrases sequences with a fixed beam depth, and then perform keyphrase ranking to select top-k keyphrases as output. Yuan et al. (2020) proposed catSeq with ONE2SEQ paradigm by adding a special token $\langle eos \rangle$ at the end to terminate the decoding process. They further introduce catSeqD by maximizing mutual information between all the keyphrases and source text and using orthogonal constraints (Bousmalis et al., 2016) to ensure the coverage and diversity of the generated keyphrase. Many works are conducted based on the ONE2SEQ paradigm (Chen et al., 2019a; Chan et al., 2019; Chen et al., 2020; Meng et al., 2021; Luo et al., 2020). Chen et al. (2019a) proposed to use the keyphrases of retrieved documents as an external input. However, the keyphrase alone lacks semantic information, and the potential knowledge in the retrieved documents are also ignored. In contrast, our method makes full use of both retrieved documents and keyphrases as references. Since catSeq tends to generate shorter sequences, a reinforcement learning approach is introduced by Chan et al. (2019) to encourage their model to generate the correct number of keyphrases with an adaptive reward (i.e., F_1 and $Recall$). More recently, Luo et al. (2021) introduced a two-stage reinforcement learning-based fine-tuning approach with a fine-grained reward score, which also considers the semantic similarities between predictions and targets. Ye et al. (2021) proposed a ONE2SET paradigm to predict the keyphrases as a set, which eliminates the bias caused by the predefined order

in ONE2SEQ paradigm. Our method can also be integrated into these methods to further improve performance, as shown in section 4.4.

5.2 Heterogeneous Graph for NLP

Different from homogeneous graph that only considers a single type of nodes or links, heterogeneous graph can deal with multiple types of nodes or links (Shi et al., 2016). Linmei et al. (2019) constructed a topic-entity heterogeneous neural graph for semi-supervised short text classification. Tu et al. (2019) introduced a heterogeneous graph neural network to encode documents, entities, and candidates together for multi-hop reading comprehension. Wang et al. (2020) presented heterogeneous graph neural network with words, sentences, and documents nodes for extractive summarization. In our paper, we study the keyword-document heterogeneous graph network for keyphrase generation, which has not been explored before.

6 Conclusions

In this paper, we propose a graph-based method that can capture explicit knowledge from related references. Our model consists of a heterogeneous graph encoder to model different granularity of relations among the source document and its references, and a hierarchical attention and copy mechanism to guide the decoding process. Extensive experiments demonstrate the effectiveness and portability of our method on both the present and absent keyphrase predictions.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by China National Key R&D Program (No. 2017YFB1002104), National Natural Science Foundation of China (No. 61976056, 62076069), Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 343–351.

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural keyphrase generation via reinforcement learning with adaptive rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2163–2174, Florence, Italy. Association for Computational Linguistics.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4057–4066, Brussels, Belgium. Association for Computational Linguistics.

Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. 2019a. An integrated approach for keyphrase generation via exploring the power of retrieval and extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2846–2856, Minneapolis, Minnesota. Association for Computational Linguistics.

Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive hierarchical decoding for deep keyphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1105, Online. Association for Computational Linguistics.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. 2019b. Title-guided encoding for keyphrase generation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6268–6275. AAAI Press.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, 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)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings*

of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2270–2282, Online. Association for Computational Linguistics.

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. AAAI Press.

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 (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223.

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

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

Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4821–4830, Hong Kong, China. Association for Computational Linguistics.

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*, pages 135–144, Portland, Oregon, USA. Association for Computational Linguistics.

Yichao Luo, Zhengyan Li, Bingning Wang, Xiaoyu Xing, Qi Zhang, and Xuanjing Huang. 2020. Sensenet: Neural keyphrase generation with document structure. In *arXiv*.

Yichao Luo, Yige Xu, Jiacheng Ye, Xipeng Qiu, and Qi Zhang. 2021. Keyphrase generation with fine-grained evaluation-guided reinforcement learning. *arXiv preprint arXiv:2104.08799*.

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*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

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*, pages 1318–1327, Singapore. Association for Computational Linguistics.

Rui Meng, Xingdi Yuan, Tong Wang, Sanqiang Zhao, Adam Trischler, and Daqing He. 2021. An empirical study on neural keyphrase generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4985–5007, Online. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 582–592, Vancouver, Canada. Association for Computational Linguistics.

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

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*.

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 (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2016. A survey of heterogeneous information network analysis. In *TKDE*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. 2019. Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs. In

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2704–2713, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219, Online. Association for Computational Linguistics.

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*, pages 4142–4153, Brussels, Belgium. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 4598–4608, Online. Association for Computational Linguistics.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. One size does not fit all: Generating and evaluating variable number of keyphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7961–7975, Online. Association for Computational Linguistics.

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*, pages 836–845, Austin, Texas. Association for Computational Linguistics.

A Results on Krapivin Dataset

Model	Krapivin			
	Present		Absent	
	$F1@5$	$F1@10$	$R@10$	$R@50$
CopyRNN (Meng et al., 2017)	0.334	0.326	0.113	0.202
CorrRNN (Chen et al., 2018)	0.358	0.330	0.108	-
TG-Net (Chen et al., 2019b)	0.406	0.370	0.146	0.253
KG-KE-KR-M (Chen et al., 2019a)	0.431	0.378	0.153	0.251
CopyRNN-GATER (Ours)	$\mathbf{0.435}_3$	$\mathbf{0.383}_2$	$\mathbf{0.195}_3$	$\mathbf{0.294}_3$

Model	Krapivin			
	Present		Absent	
	$F1@5$	$F1@M$	$F1@5$	$F1@M$
catSeq (Yuan et al., 2020)	0.269	0.354	0.018	0.036
catSeqD (Yuan et al., 2020)	0.264	0.349	0.018	0.037
catSeqCorr (Chan et al., 2019)	0.265	0.349	0.020	0.038
catSeqTG (Chan et al., 2019)	**0.282**	0.366	0.018	0.034
SenSeNet (Luo et al., 2020)	0.279	0.354	0.024	0.046
catSeq-GATER (Ours)	0.276_3	$\mathbf{0.376}_4$	$\mathbf{0.037}_3$	$\mathbf{0.069}_5$

Table 5: Keyphrase prediction results of the models trained under ONE2ONE and ONE2SEQ paradigms. The best results are bold. The subscripts are the corresponding standard deviation (e.g., 0.069_5 means 0.069 ± 0.005).