Exclusive Hierarchical Decoding for Deep Keyphrase Generation

Wang Chen[1], **Hou Pong Chan**[1], **Piji Li**[2], **Irwin King**[1]
[1]The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2]Tencent AI Lab
[1]{wchen, hpchan, king}@cse.cuhk.edu.hk
[2]pijili@tencent.com

Abstract

Keyphrase generation (KG) aims to summarize the main ideas of a document into a set of keyphrases. A new setting is recently introduced into this problem, in which, given a document, the model needs to predict a set of keyphrases and simultaneously determine the appropriate number of keyphrases to produce. Previous work in this setting employs a sequential decoding process to generate keyphrases. However, such a decoding method ignores the intrinsic hierarchical compositionality existing in the keyphrase set of a document. Moreover, previous work tends to generate duplicated keyphrases, which wastes time and computing resources. To overcome these limitations, we propose an exclusive hierarchical decoding framework that includes a hierarchical decoding process and either a soft or a hard exclusion mechanism. The hierarchical decoding process is to explicitly model the hierarchical compositionality of a keyphrase set. Both the soft and the hard exclusion mechanisms keep track of previously-predicted keyphrases within a window size to enhance the diversity of the generated keyphrases. Extensive experiments on multiple KG benchmark datasets demonstrate the effectiveness of our method to generate less duplicated and more accurate keyphrases[1].

1 Introduction

Keyphrases are short phrases that indicate the core information of a document. As shown in Figure 1, the keyphrase generation (KG) problem focuses on automatically producing a *keyphrase set* (a set of keyphrases) for the given document. Because of the condensed expression, keyphrases can benefit various downstream applications including opinion mining (Berend, 2011; Wilson et al., 2005), document clustering (Hulth and Megyesi, 2006), and text summarization (Wang and Cardie, 2013).

Input Document: … A noninvasive diagnostic device was developed to assess the vascular origin and severity of penile dysfunction. It was designed and studied using both a mathematical model of penile hemodynamics and preliminary experiments on healthy young volunteers. … Simulations using a mathematical model show that the device is capable of differentiating between arterial insufficiency and venous leak and indicate the severity of each. …

Keyphrases:
{erectile dysfunction; arterial insufficiency; venous leak; veno-occlusive mechanism; mathematical model; hemodynamics}

Figure 1: An example of an input document and its expected keyphrase output for keyphrase generation problem. Present keyphrases that appear in the document are underlined.

Keyphrases of a document can be categorized into two groups: *present keyphrase* that appears in the document and *absent keyphrase* that does not appear in the document. Recent generative methods for KG apply the attentional encoder-decoder framework (Luong et al., 2015; Bahdanau et al., 2014) with copy mechanism (Gu et al., 2016; See et al., 2017) to predict both present and absent keyphrases. To generate multiple keyphrases for an input document, these methods first use beam search to generate a huge number of keyphrases (e.g., 200) and then pick the top N ranked keyphrases as the final prediction. Thus, in other words, these methods can only predict a fixed number of keyphrases for all documents.

However, in a practical situation, the appropriate number of keyphrases varies according to the content of the input document. To simultaneously predict keyphrases and determine the suitable number of keyphrases, Yuan et al. (2018) adopts a sequential decoding method with greedy search to generate one sequence consisting of the predicted keyphrases and separators. For example, the produced sequence may be "hemodynamics [sep] erectile dysfunction [sep] ...", where "[sep]" is the sep-

[1]Our code is available at https://github.com/Chen-Wang-CUHK/ExHiRD-DKG.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1095–1105

arator. After producing an ending token, the decoding process terminates. The final keyphrase predictions are obtained after splitting the sequence by separators. However, there are two drawbacks to this method. First, the sequential decoding method ignores the hierarchical compositionality existing in a keyphrase set (a keyphrase set is composed of multiple keyphrases and each keyphrase consists of multiple words). In this work, we examine the hypothesis that a generative model can predict more accurate keyphrases by incorporating the knowledge of the hierarchical compositionality in the decoder architecture. Second, the sequential decoding method tends to generate duplicated keyphrases. It is simple to design specific post-processing rules to remove the repeated keyphrases, but generating and then removing repeated keyphrases wastes time and computing resources. To address these two limitations, we propose a novel exclusive hierarchical decoding framework for KG, which includes a hierarchical decoding process and an exclusion mechanism.

Our hierarchical decoding process is designed to explicitly model the hierarchical compositionality of a keyphrase set. It is composed of phrase-level decoding (PD) and word-level decoding (WD). A PD step determines which aspect of the document to summarize based on both the document content and the aspects summarized by previously-generated keyphrases. The hidden representation of the captured aspect is employed to initialize the WD process. Then, a new WD process is conducted under the PD step to generate a new keyphrase word by word. Both PD and WD repeat until meeting the stop conditions. In our method, both PD and WD attend the document content to gather contextual information. Moreover, the attention score of each WD step is rescaled by the corresponding PD attention score. The purpose of the attention rescaling is to indicate which aspect is focused on by the current PD step.

We also propose two kinds of exclusion mechanisms (i.e., a soft one and a hard one) to avoid generating duplicated keyphrases. Either the soft one or the hard one is used in our hierarchical decoding process. Both of them are used in the WD process of our hierarchical decoding. Besides, both of them collect the previously-generated K keyphrases, where K is a predefined window size. The soft exclusion mechanism is incorporated in the training stage, where an exclusive loss is employed to encourage the model to generate a different first word of the current keyphrase with the first words of the collected K keyphrases. However, the hard exclusion mechanism is used in the inference stage, where an exclusive search is used to force WD to produce a different first word with the first words of the collected K keyphrases. Our motivation is from the statistical observation that in 85% of the documents on the largest KG benchmark, the keyphrases of each individual document have different first words. Moreover, since a keyphrase is usually composed of only two or three words, the predicted first word significantly affects the prediction of the following keyphrase words. Thus, our exclusion mechanisms can boost the diversity of the generated keyphrases. In addition, generating fewer duplications will also improve the chance to produce correct keyphrases that have not been predicted yet.

We conduct extensive experiments on four popular real-world benchmarks. Empirical results demonstrate the effectiveness of our hierarchical decoding process. Besides, both the soft and the hard exclusion mechanisms significantly reduce the number of duplicated keyphrases. Furthermore, after employing the hard exclusion mechanism, our model consistently outperforms all the SOTA sequential decoding baselines on the four benchmarks.

We summarize our main contributions as follows: (1) to our best knowledge, we are the first to design a hierarchical decoding process for the keyphrase generation problem; (2) we propose two novel exclusion mechanisms to avoid generating duplicated keyphrases as well as improve the generation accuracy; and (3) our method consistently outperforms all the SOTA sequential decoding methods on multiple benchmarks under the new setting.

2 Related Work

2.1 Keyphrase Extraction

Most of the traditional extractive methods (Witten et al., 1999; Mihalcea and Tarau, 2004) focus on extracting present keyphrases from the input document and follow a two-step framework. They first extract plenty of keyphrase candidates by hand-crafted rules (Medelyan et al., 2009). Then, they score and rank these candidates based on either unsupervised methods (Mihalcea and Tarau, 2004) or supervised learning methods (Nguyen and Kan, 2007; Hulth, 2003). Recently, neural-based se-

quence labeling methods (Gollapalli et al., 2017; Luan et al., 2017; Zhang et al., 2016) are also explored in keyphrase extraction problem. However, these extractive methods cannot predict absent keyphrase which is also an essential part of a keyphrase set.

2.2 Keyphrase Generation

To produce both present and absent keyphrases, Meng et al. (2017) introduced a generative model, CopyRNN, which is based on an attentional encoder-decoder framework (Bahdanau et al., 2014) incorporating with a copy mechanism (Gu et al., 2016). A wide range of extensions of CopyRNN are recently proposed (Chen et al., 2018, 2019b; Ye and Wang, 2018; Chen et al., 2019a; Zhao and Zhang, 2019). All of them rely on beam search to over-generate lots of keyphrases with large beam size and then select the top N (e.g., five or ten) ranked ones as the final prediction. That means these over-generated methods will always predict N keyphrases for any input documents. Nevertheless, in a real situation, the keyphrase number should be determined by the document content and may vary among different documents.

To this end, Yuan et al. (2018) introduced a new setting that the KG model should predict multiple keyphrases and simultaneously decide the suitable keyphrase number for the given document. Two models with a sequential decoding process, catSeq and catSeqD, are proposed in Yuan et al. (2018). The catSeq is also an attentional encoder-decoder model (Bahdanau et al., 2014) with copy mechanism (See et al., 2017), but adopting new training and inference setup to fit the new setting. The catSeqD is an extension of catSeq with orthogonal regularization (Bousmalis et al., 2016) and target encoding. Lately, Chan et al. (2019) proposed a reinforcement learning based fine-tuning method, which fine-tunes the pre-trained models with adaptive rewards for generating more sufficient and accurate keyphrases. We follow the same setting with Yuan et al. (2018) and propose an exclusive hierarchical decoding method for the KG problem. To the best of our knowledge, this is the first time the hierarchical decoding is explored in the KG problem. Different from the hierarchical decoding in other areas (Fan et al., 2018; Yarats and Lewis, 2018; Tan et al., 2017; Chen and Zhuge, 2018), we rescale the attention score of each WD step with the corresponding PD attention score to provide aspect guidance when generating keyphrases. Moreover, either a soft or a hard exclusion mechanism is innovatively incorporated in the decoding process to improve generation diversity.

3 Notations and Problem Definition

We denote vectors and matrices with bold lowercase and uppercase letters respectively. Sets are denoted with calligraphy letters. We use $\mathbf{W}$ to represent a parameter matrix.

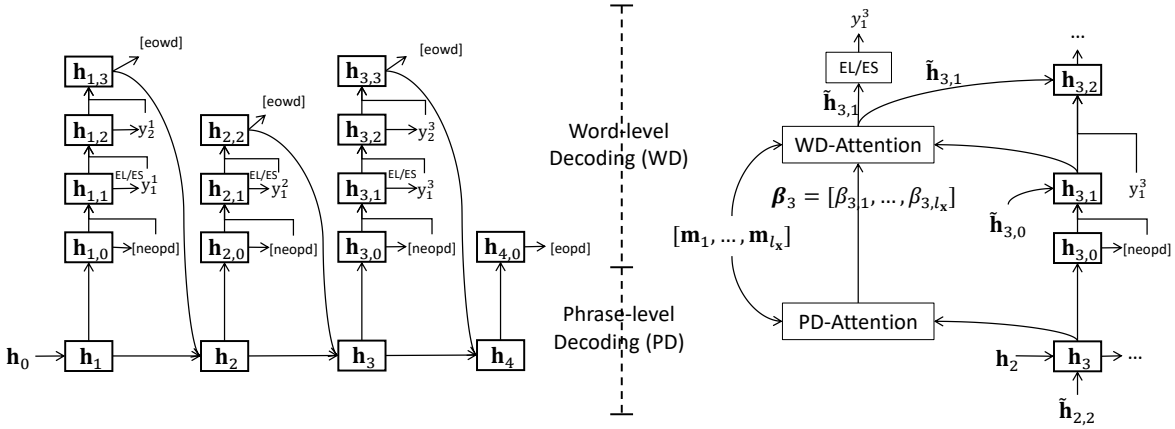
We define the keyphrase generation problem as follows. The input is a document $\mathbf{x}$, the output is a keyphrase set $\mathcal{Y} = \{\mathbf{y}^i\}_{i=1,...,|\mathcal{Y}|}$, where $|\mathcal{Y}|$ is the keyphrase number of $\mathbf{x}$. Both the $\mathbf{x}$ and each $\mathbf{y}^i$ are sequences of words, i.e., $\mathbf{x} = [x_1, ..., x_{l_\mathbf{x}}]$ and $\mathbf{y}^i = [y^i_1, ..., y^i_{l_{\mathbf{y}^i}}]$, where $l_\mathbf{x}$ and $l_{\mathbf{y}^i}$ are the word numbers of $\mathbf{x}$ and $\mathbf{y}^i$ correspondingly.

4 Our Methodology

We first encode each word of the document into a hidden state and then employ our exclusive hierarchical decoding shown in Figure 2 to produce keyphrases for the given document. Our hierarchical decoding process consists of phrase-level decoding (PD) and word-level decoding (WD). Each PD step decides an appropriate aspect to summarize based on both the context of the document and the aspects summarized by previous PD steps. Then, the hidden representation of the captured aspect is employed to initialize the WD process to generate a new keyphrase word by word. The WD process terminates when producing a "[eowd]" token. If the WD process output a "[eopd]" token, the whole hierarchical decoding process stops. Both PD and WD attend the document content. The PD attention score is used to re-weight the WD attention score to provide aspect guidance. To improve the diversity of the predicted keyphrases, we incorporate either an exclusive loss when training (i.e., the soft exclusion mechanism) or an exclusive search mechanism when inference (i.e., the hard exclusion mechanism).

4.1 Sequential Encoder

To obtain the context-aware representation of each document word, we employ a two-layered bidirectional GRU (Cho et al., 2014) as the document encoder: $\mathbf{m}_k = \text{BiGRU}(\mathbf{e}_{x_k}, \overrightarrow{\mathbf{m}}_{k-1}, \overleftarrow{\mathbf{m}}_{k+1})$, where $k = 1, 2, ..., l_\mathbf{x}$ and $\mathbf{e}_{x_k}$ is the embedding vector of x_k with d_e dimensions. $\mathbf{m}_k = [\overrightarrow{\mathbf{m}}_k; \overleftarrow{\mathbf{m}}_k] \in \mathbb{R}^d$



(a) The simplified framework of our exclusive hierarchical decoding (b) The intermediate step to predict y_1^3

Figure 2: Illustration of our exclusive hierarchical decoding. $\mathbf{h}_i$ is the hidden state of i-th PD step. $\mathbf{h}_{i,j}$ is the corresponding j-th WD hidden state. The "[neopd]" token means PD does not end. The "[eowd]" token means WD terminates. The "[eopd]" token means PD ends and the whole decoding process finishes. "$[\mathbf{m}_1, \ldots, \mathbf{m}_{l_\mathbf{x}}]$" represents the encoded hidden states from the document. "PD-Attention" and "WD-Attention" are the attention mechanisms in PD and WD respectively. "$\boldsymbol{\beta}_i$" is the PD attention score at i-th step. $\tilde{\mathbf{h}}_{i,j}$ is the WD attentional vector. "EL/ES" indicates either the exclusive loss or the exclusive search is incorporated.

is the encoded context-aware representation of x_k. Here, "$[\cdot\,;\cdot]$" means concatenation.

4.2 Hierarchical Decoder

Our hierarchical decoding process is controlled by the hierarchical decoder, which utilizes a phrase-level decoder and a word-level decoder to handle the PD process and the WD process respectively. We present our hierarchical decoder first and then introduce the exclusion mechanisms. In our decoders, all the hidden states and attentional vectors are d-dimensional vectors.

4.2.1 Phrase-level Decoder

We adopt a unidirectional GRU layer as our phrase-level decoder. After the WD process under last PD step is finished, the phrase-level decoder will update its hidden state as follows:

$$\mathbf{h}_i = \overrightarrow{\text{GRU}}_1(\tilde{\mathbf{h}}_{i-1,end}, \mathbf{h}_{i-1}), \tag{1}$$

where $\tilde{\mathbf{h}}_{i-1,end}$ is the attentional vector for the ending WD step under the (i-1)-th PD step (e.g., $\tilde{\mathbf{h}}_{2,2}$ in Figure 2(b)). $\mathbf{h}_i$ is regarded as the hidden representation of the captured aspect at the i-th PD step. $\mathbf{h}_0$ is initialized as the document representation $[\overrightarrow{\mathbf{m}}_{l_\mathbf{x}}; \overleftarrow{\mathbf{m}}_1]$. $\tilde{\mathbf{h}}_{0,end}$ is initialized with zeros.

In PD-Attention process, the PD attentional score $\boldsymbol{\beta}_i = [\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,l_\mathbf{x}}]$ is computed from the following attention mechanism employing $\mathbf{h}_i$ as the query vector:

$$\beta_{i,k} = \exp(s_{i,k}) / \sum_{n=1}^{l_\mathbf{x}} \exp(s_{i,n}), \tag{2}$$

$$s_{i,n} = (\mathbf{h}_i)^T \mathbf{W}_1 \mathbf{m}_n. \tag{3}$$

4.2.2 Word-level Decoder

We choose another unidirectional GRU layer to conduct word-level decoding. Under the i-th PD step, the word-level decoder updates its hidden state first:

$$\mathbf{h}_{i,j} = \overrightarrow{\text{GRU}}_2([\tilde{\mathbf{h}}_{i,j-1}; \mathbf{e}_{y^i_{j-1}}], \mathbf{h}_{i,j-1}), \tag{4}$$

where $\tilde{\mathbf{h}}_{i,j-1}$ is the WD attentional vector of the (j-1)-th WD step and $\mathbf{e}_{y^i_{j-1}}$ is the d_e-dimensional embedding vector of the y^i_{j-1} token. We define $\mathbf{h}_{i,0} = \overrightarrow{\text{GRU}}_2([\mathbf{0}; \mathbf{e}_s], \mathbf{h}_i)$, where $\mathbf{h}_i$ is the current hidden state of the phrase-level decoder, $\mathbf{0}$ is a zero vector, and $\mathbf{e}_s$ is the embedding of the start token. Then, the WD attentional vector is computed:

$$\tilde{\mathbf{h}}_{i,j} = \tanh(\mathbf{W}_2[\mathbf{h}_{i,j}; \mathbf{a}_{i,j}]), \tag{5}$$

$$\mathbf{a}_{i,j} = \sum_{k=1}^{l_\mathbf{x}} \bar{\alpha}_{(i,j),k} \mathbf{m}_k, \tag{6}$$

$$\bar{\alpha}_{(i,j),k} = \frac{\alpha_{(i,j),k} \times \beta_{i,k}}{\sum_{n=1}^{l_\mathbf{x}} \alpha_{(i,j),n} \times \beta_{i,n}}, \tag{7}$$

where $\alpha_{(i,j),k}$ is the original WD attention score which is computed similar to $\beta_{i,k}$ except that a

new parameter matrix is used and $\mathbf{h}_{i,j}$ is employed as the query vector. The purpose of the rescaling operation in Eq. (7) is to indicate the focused aspect of the current PD step for each WD step.

Finally, the $\tilde{\mathbf{h}}_{i,j}$ is utilized to predict the probability distribution of current keyword with the copy mechanism (See et al., 2017):

$$P_j^i = (1 - g_j^i)P_{j,\mathcal{V}}^i + g_j^i P_{j,\mathcal{X}}^i, \quad (8)$$

where $g_j^i = \text{sigmoid}(\mathbf{w}_g^T \tilde{\mathbf{h}}_{i,j} + b_g) \in \mathbb{R}$ is the copy gate. $P_{j,\mathcal{V}}^i = \text{softmax}(\mathbf{W}_3 \tilde{\mathbf{h}}_{i,j} + \mathbf{b}_\mathcal{V}) \in \mathbb{R}^{|\mathcal{V}|}$ is the probability distribution over a predefined vocabulary $\mathcal{V}$. $P_{j,\mathcal{X}}^i = \sum_{k:x_k=y_j^i} \bar{\alpha}_{(i,j),k} \in \mathbb{R}^{|\mathcal{X}|}$ is the copying probability distribution over $\mathcal{X}$ which is a set of all the words that appeared in the document. $P_j^i \in \mathbb{R}^{|\mathcal{V} \cup \mathcal{X}|}$ is the final predicted probability distribution. Finally, greedy search is applied to produce the current token.

The WD process terminates when producing a "[eowd]" token. The whole hierarchical decoding process ends if the word-level decoder produces a "[eopd]" token at the 0-th step, i.e., y_0^i is predicted as "[eopd]".

4.3 Training

A standard negative log-likelihood loss is employed as the generation loss to train our hierarchical decoding model:

$$\mathcal{L}_g = -\sum_{i=1}^{|\bar{\mathcal{Y}}|} \sum_{j=0}^{l_{\bar{\mathbf{y}}^i}} \log P_j^i(\bar{y}_j^i | \mathbf{x}; \bar{\mathbf{Y}}^{i-1}; \bar{\mathbf{y}}_{j-1}^i), \quad (9)$$

where $\bar{\mathbf{Y}}^{i-1} = \bar{\mathbf{y}}^1, \ldots, \bar{\mathbf{y}}^{i-1}$ are the target keyphrases of previously-finished PD steps and $\bar{\mathbf{y}}_{j-1}^i = \bar{y}_0^i, \ldots, \bar{y}_{j-1}^i$ are target keyphrase words of previous WD steps under the i-th PD step. When training, each original target keyphrase is extended with a "[neopd]" token and a "[eowd]" token, i.e., $\bar{\mathbf{y}}^i =$ ["[neopd]", $y_1^i, \ldots, y_{l_{\mathbf{y}^i}}^i$, "[eowd]"]. Besides, a "[eopd]" token is also incorporated into the targets to indicate the ending of whole decoding process. Teacher forcing is employed when training.

4.4 Soft and Hard Exclusion Mechanisms

To alleviate the duplication generation problem, we propose a soft and a hard exclusion mechanisms. Either of them can be incorporated into our hierarchical decoding process to form one kind of exclusive hierarchical decoding method.

Soft Exclusion Mechanism. An exclusive loss (EL) is introduced in the training stage as shown in Algorithm 1. "$j == 1$" in line "3" means the current WD step is predicting the first word of a keyphrase. In short, the exclusive loss punishes the model for the tendency to generate the same first word of the current keyphrase with the first words of previously-generated keyphrases within the window size K_{EL}.

Algorithm 1 Training with Exclusive Loss

Require: The window size K_{EL}. The target keyphrases $[\bar{\mathbf{y}}^1, \ldots, \bar{\mathbf{y}}^i, \ldots, \bar{\mathbf{y}}^{|\bar{\mathcal{Y}}|}]$. The predicted probability distribution P_j^i for the j-th WD step under the i-th PD step where $i = 1, \ldots, |\bar{\mathcal{Y}}|$ and $j = 0, 1, \ldots, l_{\bar{\mathbf{y}}^i}$.

1: Firstly, the exclusive loss of the j-th WD step under the i-th PD step is computed as follows.
2: $K_{EL} \leftarrow \min\{K_{EL}, i-1\}$
3: **if** $K_{EL} > 0$ **and** $j == 1$ **then**
4: $\quad \mathcal{L}_{EL}^{i,j} = \sum_{idx=i-K_{EL}, \bar{y}_j^{idx} \neq \bar{y}_j^i}^{i-1} -\log(1 - P_j^i(\bar{y}_j^{idx}))$
5: **else**
6: $\quad \mathcal{L}_{EL}^{i,j} = 0.0$
7: **end if**
8: Secondly, the exclusive loss for the whole decoding process is calculated as $\mathcal{L}_{EL} = \sum_{i,j} \mathcal{L}_{EL}^{i,j}$.
9: Finally, the joint loss $\mathcal{L} = \mathcal{L}_g + \mathcal{L}_{EL}$ is employed to train the model.

Algorithm 2 Inference with Exclusive Search

Require: The window size K_{ES}. The first words of previously-predicted keyphrases $[y_1^1, \ldots, y_1^{i-1}]$. The current WD step index j. The predicted probability distribution P_j^i for current WD step.

$K_{ES} \leftarrow \min\{K_{ES}, i-1\}$
if $K_{ES} > 0$ **and** $j == 1$ **then**
$\quad$ **for** $idx = i - K_{ES}, i - K_{ES} + 1, \ldots, i - 1$ **do**
$\quad\quad P_j^i(y_j^{idx}) \leftarrow 0.0$
$\quad$ **end for**
end if
Return $y_j^i = \arg\max(P_j^i)$ as the predicted word for current WD step.

Hard Exclusion Mechanism. An exclusive search (ES) is introduced in the inference stage as shown in Algorithm 2. The exclusive search mechanism forces the word-level decoding to predict a different first word with the first words of previously-predicted keyphrases within the window size K_{ES}.

Since a keyphrase usually has only two or three words, the first word significantly affects the prediction of the following words. Therefore, both the soft and the hard exclusion mechanisms can improve the diversity of generated keyphrases.

5 Experiment Setup

Our model implementations are based on the OpenNMT system (Klein et al., 2017) using PyTorch (Paszke et al., 2017). Experiments of all

models are repeated with three different random seeds and the averaged results are reported.

5.1 Datasets

We employ four scientific article benchmark datasets to evaluate our models, including **KP20k** (Meng et al., 2017), **Inspec** (Hulth, 2003), **Krapivin** (Krapivin et al., 2009), and **SemEval** (Kim et al., 2010). Following previous work (Yuan et al., 2018; Chen et al., 2019a), we use the training set of KP20k to train all the models. After removing the duplicated data, we maintain 509,818 data samples in the training set, 20,000 in the validation set, and 20,000 in the testing set. After training, we test all the models on the testing datasets of these four benchmarks. The dataset statistics are shown in Table 1.

Dataset	**Total**	**Validation**	**Testing**
Inspec	2,000	1,500	500
Krapivin	2,303	1,903	400
SemEval	244	144	100
KP20k	549,818	20,000	20,000

Table 1: The statistics of validation and testing datasets.

5.2 Baselines

We focus on the comparisons with state-of-the-art decoding methods and choose the following generation models under the new setting as our baselines:

- **Transformer** (Vaswani et al., 2017). A transformer-based sequence to sequence model incorporating with copy mechanism.
- **catSeq** (Yuan et al., 2018). An RNN-based attentional encoder-decoder model with copy mechanism. Both the encoding and decoding are sequential.
- **catSeqD** (Yuan et al., 2018). An extension of catSeq which incorporates orthogonal regularization (Bousmalis et al., 2016) and target encoding into the sequential decoding process to improve the generation diversity and accuracy.
- **catSeqCorr** (Chan et al., 2019). Another extension of catSeq, which incorporates the sequential decoding with coverage (See et al., 2017) and review mechanisms to boost the generation diversity and accuracy. This method is adjusted from Chen et al. (2018) to fit the new setting.

In this paper, we propose two novel models that are denoted as follows:

- **ExHiRD-s**. Our **Ex**clusive **Hie**ra**R**chical **D**ecoding model with the **s**oft exclusion mechanism. In experiments, the window size K_{EL} is selected as 4 after tuning on the KP20k validation dataset.
- **ExHiRD-h**. Our **Ex**clusive **Hie**ra**R**chical **D**ecoding model with the **h**ard exclusion mechanism. In experiments, the values of the window size K_{ES} are selected as 4, 1, 1, 1 for Inspec, Krapivin, SemEval, and KP20k respectively after tuning on the corresponding validation datasets.

We choose the bilinear attention from Luong et al. (2015) and the copy mechanism from See et al. (2017) for all the models.

5.3 Evaluation Metrics

We engage $F_1@M$ which is recently proposed in Yuan et al. (2018) as one of our evaluation metrics. $F_1@M$ compares all the predicted keyphrases by the model with ground-truth keyphrases, which means it does not use a fixed cutoff for the predictions. Therefore, it considers the number of predictions.

We also use $F_1@5$ as another evaluation metric. When the number of predictions is less than five, we randomly append incorrect keyphrases until it obtains five predictions instead of directly using the original predictions. If we do not adopt such an appending operation, $F_1@5$ will become the same with $F_1@M$ when the prediction number is less than five.

The macro-averaged $F_1@M$ and $F_1@5$ scores are reported. When determining whether two keyphrases are identical, all the keyphrases are stemmed first. Besides, all the duplicated keyphrases are removed after stemming.

5.4 Implementation Details

Following previous work (Meng et al., 2017; Yuan et al., 2018; Chen et al., 2019a; Chan et al., 2019), we lowercase the characters, tokenize the sequences, and replace digits with "<digit>" token. Similar to Yuan et al. (2018), when training, the present keyphrase targets are sorted according to the orders of their first occurrences in the document. Then, the absent keyphrase targets are put at the end of the sorted present keyphrase targets. We use "<p_start>" and "<a_start>" as the

Model	Inspec		Krapivin		SemEval		KP20k	
	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$
Transformer	0.254_{5}	0.210_{7}	0.328_{14}	0.252_{4}	0.310_{5}	0.257_{4}	0.360_{3}	0.282_{10}
catSeq	0.276_{5}	0.233_{4}	0.344_{14}	0.269_{5}	0.313_{8}	0.262_{11}	0.368_{1}	0.295_{2}
catSeqD	0.280_{3}	0.236_{1}	0.344_{9}	0.268_{8}	0.311_{6}	0.263_{6}	0.368_{2}	0.296_{2}
catSeqCorr	0.253_{3}	0.208_{6}	0.343_{9}	0.258_{9}	0.318_{18}	0.260_{14}	0.367_{3}	0.281_{4}
ExHiRD-s	0.278_{5}	0.235_{3}	0.338_{3}	0.278_{0}	0.322_{5}	0.276_{5}	0.372_{1}	0.307_{0}
ExHiRD-h	$\mathbf{0.291_{3}}$	$\mathbf{0.253_{4}}$	$\mathbf{0.347_{4}}$	$\mathbf{0.286_{4}}$	$\mathbf{0.335_{17}}$	$\mathbf{0.284_{15}}$	$\mathbf{0.374_{0}}$	$\mathbf{0.311_{1}}$

Table 2: Present keyphrase prediction results of all models on all datasets. The best results are bold. In all the tables of this paper, the subscript represents the corresponding standard deviation (e.g., 0.311_1 indicates 0.311 ± 0.001).

Model	Inspec		Krapivin		SemEval		KP20k	
	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$
Transformer	0.013_{1}	0.006_{1}	0.030_{5}	0.014_{3}	0.020_{1}	0.013_{1}	0.024_{2}	0.011_{1}
catSeq	0.008_{3}	0.004_{1}	0.033_{4}	0.015_{2}	0.017_{2}	0.012_{1}	0.023_{1}	0.010_{0}
catSeqD	0.010_{4}	0.004_{1}	0.033_{7}	0.015_{3}	0.016_{1}	0.011_{1}	0.023_{1}	0.010_{1}
catSeqCorr	0.007_{2}	0.004_{1}	0.022_{6}	0.011_{3}	0.021_{5}	0.014_{3}	0.023_{1}	0.010_{1}
ExHiRD-s	0.021_{7}	0.009_{2}	0.033_{5}	0.016_{2}	0.024_{5}	0.016_{4}	0.029_{1}	0.014_{0}
ExHiRD-h	$\mathbf{0.022_{3}}$	$\mathbf{0.011_{1}}$	$\mathbf{0.043_{6}}$	$\mathbf{0.022_{3}}$	$\mathbf{0.025_{6}}$	$\mathbf{0.017_{4}}$	$\mathbf{0.032_{0}}$	$\mathbf{0.016_{0}}$

Table 3: Absent keyphrase prediction results of all models on all datasets. The best results are bold.

"[neopd]" token of present and absent keyphrases respectively. ";" is employed as the "[eowd]" token for both present and absent keyphrases. "</s>" is used as the "[eopd]" token.

The vocabulary with 50,000 tokens is shared between the encoder and decoder. We set d_e as 100 and d as 300. The hidden states of the encoder layers are initialized as zeros. In the training stage, we randomly initialize all the trainable parameters including the embedding using a uniform distribution in $[-0.1, 0.1]$. We set batch size as 10, max gradient norm as 1.0, and initial learning rate as 0.001. We do not use dropout. Adam (Kingma and Ba, 2014) is used as our optimizer. The learning rate decays to half if the perplexity on KP20k validation set stops decreasing. Early stopping is applied when training. When inference, we set the minimum phrase-level decoding step as 1 and the maximum as 20.

6 Results and Analysis

6.1 Present and Absent Keyphrase Predictions

We show the present and absent keyphrase prediction results in Table 2 and Table 3 correspondingly. As indicated in these two tables, both the ExHiRD-s model and the ExHiRD-h outperform the state-of-the-art baselines on most of the metrics, which demonstrates the effectiveness of our exclusive hierarchical decoding methods. Besides, the ExHiRD-h model consistently achieves the best results on both present and absent keyphrase prediction in all the datasets[2].

Model	Inspec	Krapivin	SemEval	KP20k
Transformer	0.286_{25}	0.297_{46}	0.220_{38}	0.223_{41}
catSeq	0.302_{11}	0.277_{8}	0.200_{2}	0.217_{4}
catSeqD	0.304_{14}	0.283_{9}	0.199_{1}	0.215_{8}
catSeqCorr	0.352_{38}	0.354_{4}	0.249_{23}	0.282_{14}
ExHiRD-s	0.210_{14}	0.182_{12}	0.119_{8}	0.137_{6}
ExHiRD-h	$\mathbf{0.030_{6}}$	$\mathbf{0.140_{6}}$	$\mathbf{0.091_{10}}$	$\mathbf{0.110_{1}}$

Table 4: The average DupRatios of predicted keyphrases on all datasets. The lower the score, the better the performance.

6.2 Duplication Ratio of Predicted Keyphrases

In this section, we study the model capability of avoiding producing duplicated keyphrases. Duplication ratio is denoted as "DupRatio" and defined as follows:

$$DupRatio = \frac{\#\, duplications}{\#\, predictions}, \qquad (10)$$

where # means "the number of". For instance, the DupRatio is 0.5 (3/6) for [A, A, B, B, A, C].

We report the average DupRatio per document in Table 4. From this table, we observe that our ExHiRD-s and ExHiRD-h consistently and significantly reduce the duplication ratios on all datasets. Moreover, we also find that our ExHiRD-h model achieves the lowest duplication ratios on all datasets.

[2]We also tried to simultaneously incorporate the soft and the hard exclusion mechanisms into our hierarchical decoding model, but it still underperforms ExHiRD-h.

Model	Inspec		Krapivin		SemEval		KP20k	
	#PK	#AK	#PK	#AK	#PK	#AK	#PK	#AK
Oracle	7.64	2.10	3.27	2.57	6.28	8.12	3.32	1.93
Transformer	3.17_{10}	0.70_{4}	3.57_{29}	0.63_{4}	3.24_{20}	0.67_{3}	3.44_{17}	0.58_{4}
catSeq	3.33_{2}	0.58_{4}	3.70_{10}	0.63_{5}	3.45_{5}	0.64_{3}	3.70_{4}	0.51_{2}
catSeqD	3.33_{4}	0.58_{2}	3.66_{10}	0.61_{1}	3.47_{5}	0.63_{7}	3.74_{3}	0.50_{2}
catSeqCorr	3.07_{7}	0.53_{2}	$\mathbf{3.39_{14}}$	0.56_{1}	3.15_{3}	0.62_{1}	$\mathbf{3.36_{4}}$	0.50_{1}
ExHiRD-s	3.56_{5}	0.81_{2}	4.33_{7}	0.86_{3}	$\mathbf{3.69_{14}}$	0.79_{6}	3.94_{2}	0.69_{1}
ExHiRD-h	$\mathbf{4.00_{4}}$	$\mathbf{1.50_{6}}$	4.41_{9}	$\mathbf{1.02_{7}}$	3.65_{13}	$\mathbf{0.99_{4}}$	3.97_{3}	$\mathbf{0.81_{1}}$

Table 5: Results of average numbers of predicted unique keyphrases per document. "#PK" and "#AK" are the number of present and absent keyphrases respectively. "Oracle" is the gold average keyphrase number. The closest values to the oracles are bold.

Model	Present			Absent			DupRatio
	$F_1@M$	$F_1@5$	#PK	$F_1@M$	$F_1@5$	#AK	
ExHiRD-h	**0.335**	**0.284**	**3.65**	**0.025**	**0.017**	**0.99**	**0.091**
w/o HRD	0.320	0.274	3.58	0.018	0.013	0.97	0.093
w/o ES	0.330	0.278	3.51	0.022	0.014	0.70	0.191

Table 6: Ablation study of our ExHiRD-h model on SemEval dataset. "w/o HRD" means the hierarchical decoder is replaced with a sequential decoder and the exclusive search is still incorporated. "w/o ES" represents our hierarchical decoding model without utilizing exclusive search mechanism.

6.3 Number of Predicted Keyphrases

We also study the average number of unique keyphrase predictions per document. Duplicated keyphrases are removed. The results are shown in Table 5. One main finding is that all the models generate an insufficient number of unique keyphrases on most datasets, especially for predicting absent keyphrases. We also observe that our methods can improve the number of unique keyphrases by a large margin, which is extremely beneficial to solve the problem of insufficient generation. Correspondingly, it also leads to over-generate more keyphrases than the ground-truth for the cases that do not have this problem, such as the present keyphrase predictions on Krapivin and KP20k datasets. We leave solving the over-generation of present keyphrases on Krapivin and KP20k as our future work.

6.4 ExHiRD-h: Ablation Study

Since our ExHiRD-h model achieves the best performance on almost all of the metrics, we select it as our final model and probe it more subtly in the following sections. In order to understand the effects of each component of ExHiRD-h, we conduct an ablation study on it and report the results on the SemEval dataset in Table 6.

We observe that both our hierarchical decoding process and exclusive search mechanism are helpful to generate more accurate present and absent keyphrases. Besides, we also find that the significant performance margins on the duplication ratio and the keyphrase numbers are mainly from the exclusive search mechanism.

K_{ES}	Present			Absent			DupRatio
	$F_1@M$	$F_1@5$	#PK	$F_1@M$	$F_1@5$	#AK	
Oracle	-	-	3.32	-	-	1.93	-
0	0.376	0.303	3.76	0.028	0.013	0.61	0.195
1	0.374	0.311	3.97	0.033	0.016	0.86	0.110
2	0.371	0.314	4.11	0.034	0.017	1.00	0.069
3	0.368	0.316	4.21	0.034	0.017	1.08	0.038
4	0.366	0.316	4.27	0.033	0.017	1.16	0.017
5	0.366	0.316	4.30	0.033	0.017	1.19	0.010
all	0.365	0.316	4.32	0.032	0.017	1.25	0.002

Table 7: Results of ExHiRD-h on KP20k with different window size K_{ES}. When $K_{ES} = 0$, ExHiRD-h equals to "w/o ES". The "all" means we taking the first words of all the previously-predicted keyphrases into consideration. The "DupRatio" is the average DupRatio per document. We show the average numbers of ground-truth keyphrases in the "Oracle" row.

6.5 ExHiRD-h: Window Size of Exclusive Search

For a more comprehensive understanding of our exclusive search mechanism in our ExHiRD-h model, we also study the effects of the window size K_{ES}. We conduct the experiments on KP20k dataset and list the results in Table 7.

We note that a larger window size K_{ES} leads to a lower DupRatio as we anticipated. It is because the exclusive search can observe more previously-generated keyphrases to avoid generating duplicated keyphrases when K_{ES} is larger. When K_{ES} is "all", the DupRatio is not absolute zero because we stem keyphrases when determining whether they are duplicated. Besides, we also find that larger K_{ES} leads to better $F_1@5$ scores. The reason is that for $F_1@5$ scores, we append incorrect keyphrases to obtain five predictions when the number of predictions is less than five. A larger K_{ES} leads to predict more unique keyphrases, append less absolutely incorrect keyphrases and improve the chance to output more accurate keyphrases. However, generating more unique keyphrases may also lead to more incorrect predictions, which will degrade the $F_1@M$ scores since $F_1@M$ considers all the unique predictions without a fixed cutoff.

Model	Present			Absent			DupRatio
	$F_1@M$	$F_1@5$	#PK	$F_1@M$ w	$F_1@5$	#AK	
Oracle	-	-	3.32	-	-	1.93	-
Transformer	0.360	0.282	3.44	0.024	0.011	0.58	0.223
catSeq	0.368	0.295	3.70	0.023	0.010	0.51	0.217
catSeqD	0.368	0.296	3.74	0.023	0.010	0.50	0.215
catSeqCorr	0.367	0.281	**3.36**	0.023	0.010	0.50	0.282
Transformer w/ ES	0.359	0.294	3.75	0.027	0.013	0.79	0.114
catSeq w/ ES	0.366	0.305	3.95	0.025	0.012	0.68	0.138
catSeqD w/ ES	0.366	0.306	3.99	0.026	0.012	0.65	0.137
catSeqCorr w/ ES	0.366	0.298	3.74	0.027	0.013	0.72	0.159
ExHiRD-h	**0.374**	**0.311**	3.97	**0.032**	**0.016**	**0.81**	**0.110**

Table 8: Results of applying our exclusive search to other baselines on KP20k. The "w/ ES" means our exclusive search is applied.

6.6 ExHiRD-h: Incorporate Baselines with Exclusive Search

Our exclusive search is a general method that can be easily applied to other models. In this section, we study the effects of our exclusive search on other baseline models. We show the experimental results on KP20k dataset in Table 8.

From this table, we note that the effects of exclusive search on baselines are similar to the effects on our hierarchical decoding. We also see our ExHiRD-h still achieves the best performance on most of the metrics, even if baselines are also incorporated with exclusive search, which exhibits the superiority of our hierarchical decoding again.

6.7 ExHiRD-h: Case Study

We display a prediction example in Figure 3. Our ExHiRD-h model generates more accurate keyphrases for the document comparing to the four baselines. Besides, we also observe much less repeated keyphrases are generated by our ExHiRD-h. For instance, all the baselines produce the keyphrase "debugging" at least three times. However, our ExHiRD-h only generates it once, which demonstrates that our proposed method is more powerful in avoiding duplicated keyphrases.

7 Conclusion and Future Work

In this paper, we propose an exclusive hierarchical decoding framework for keyphrase generation. Unlike previous sequential decoding methods, our hierarchical decoding consists of a phrase-level decoding process to capture the current aspect to summarize and a word-level decoding process to generate keyphrases based on the captured aspect. Besides, we also propose a soft and a hard exclusion mechanisms to enhance the diversity of the generated keyphrases. Extensive experimental results demonstrate the effectiveness of our methods. One interesting future direction is to explore whether the beam search is helpful to our model.

SOC HW/SW co-verification based debugging technique. Purpose – Increasingly complex and sophisticated VLSI design, coupled with shrinking design cycles, requires shorter verification time and efficient debug method. … SOC HW/SW co-verification technique seems to draw a balance, but Design Under Test (DUT) still resides in FPGA and remains hard for debugging. The purpose of this paper is to study a run-time RTL debugging methodology for a FPGA-based co-verification system. …

Targets {computer hardware; computer software; co-verification; debugging}

Transformer: 1. **co-verification** (1), 2. **debugging** (7), 3. fpga (3)
catSeq: 1. **debugging** (3), 2. logic programming (2)
catSeqD: 1. **debugging** (4), 2. design (3), 3. verification (3)
catSeqCorr: 1. **debugging** (3), 2. computer aided design (3)
ExHiRD-h: 1. verification (2), 2. **debugging** (1), 3. simulation (1), 4. **co-verification** (1), 5. hdl (1), 6. **computer software** (2), 7. logic testing (1)

Figure 3: An example of generated keyphrases by baselines and our ExHiRD-h. The correct predictions are bold and the present keyphrases are underlined. The digit in parentheses represents the frequency that the corresponding keyphrase is generated by the model (e.g., "debugging (3)" means the keyphrase "debugging" is generated three times by the model).

Acknowledgments

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 2300174 (Collaborative Research Fund, No. C5026-18GF)). We would like to thank our colleagues for their comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2014*.

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1162–1170, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *NeurIPS 2016*, 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.

Jingqiang Chen and Hai Zhuge. 2018. Abstractive text-image summarization using multi-modal attentional hierarchical RNN. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4046–4056, Brussels, Belgium. 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, 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.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Sujatha Das Gollapalli, Xiaoli Li, and Peng Yang. 2017. Incorporating expert knowledge into keyphrase extraction. In *AAAI 2017*, pages 3180–3187.

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.

Anette Hulth and Beáta B. Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 537–544, Sydney, Australia. Association for Computational Linguistics.

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

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 Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

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

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*, pages 2641–2651, Copenhagen, Denmark. Association for Computational Linguistics.

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, 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 *ICADL 2007*, pages 317–326.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

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.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181, Vancouver, Canada. 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, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

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 (Volume 1: Long Papers)*, pages 1395–1405, Sofia, Bulgaria. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

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 1999*, pages 254–255.

Denis Yarats and Mike Lewis. 2018. Hierarchical text generation and planning for strategic dialogue. In *ICML 2018*, pages 5587–5595.

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.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2018. One size does not fit all: Generating and evaluating variable number of 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*, pages 836–845, Austin, Texas. Association for Computational Linguistics.

Jing Zhao and Yuxiang Zhang. 2019. Incorporating linguistic constraints into keyphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5224–5233, Florence, Italy. Association for Computational Linguistics.