SGG: Learning to Select, Guide, and Generate for Keyphrase Generation

Jing Zhao, Junwei Bao, Yifan Wang, Youzheng Wu, Xiaodong He, Bowen Zhou
JD AI Research
{zhaojing857,baojunwei,wangyifan15,wuyouzheng1}@jd.com
{xiaodong.he,bowen.zhou}@jd.com

Abstract

Keyphrases, that concisely summarize the high-level topics discussed in a document, can be categorized into *present* keyphrase which explicitly appears in the source text, and *absent* keyphrase which does not match any contiguous subsequence but is highly semantically related to the source. Most existing keyphrase generation approaches synchronously generate present and absent keyphrases without explicitly distinguishing these two categories. In this paper, a **S**elect-**G**uide-**G**enerate (SGG) approach is proposed to deal with present and absent keyphrase generation *separately* with different mechanisms. Specifically, SGG is a hierarchical neural network which consists of a pointing-based selector at low layer concentrated on present keyphrase generation, a selection-guided generator at high layer dedicated to absent keyphrase generation, and a guider in the middle to transfer information from selector to generator. Experimental results on four keyphrase generation benchmarks demonstrate the effectiveness of our model, which significantly outperforms the strong baselines for both present and absent keyphrases generation. Furthermore, we extend SGG to a title generation task which indicates its extensibility in natural language generation tasks.

1 Introduction

Automatic keyphrase prediction recommends a set of representative phrases that are related to the main topics discussed in a document (Liu et al., 2009). Since keyphrases can provide a high-level topic description of a document, they are beneficial for a wide range of natural language processing (NLP) tasks, such as information extraction (Wan and Xiao, 2008), text summarization (Wang and Cardie, 2013) and question generation (Subramanian et al., 2018).

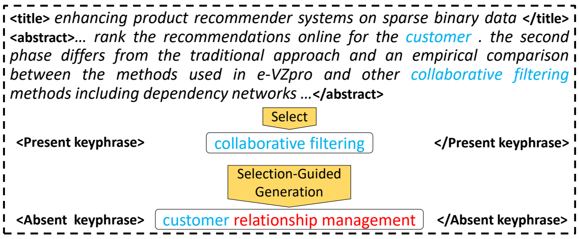

<title> *enhancing product recommender systems on sparse binary data* </title>
<abstract>*... rank the recommendations online for the customer . the second phase differs from the traditional approach and an empirical comparison between the methods used in e-VZpro and other collaborative filtering methods including dependency networks ...*</abstract>

Figure 1: An example of keyphrase prediction by SGG.

Existing methods for keyphrase prediction can be categorized into *extraction* and *generation* approaches. Specifically, keyphrase extraction methods identify important consecutive words from a given document as keyphrases, which means that the extracted keyphrases (denoted as *present keyphrases*) must exactly come from the given document. However, some keyphrases (denoted as *absent keyphrases*) of a given document do not match any contiguous subsequence but are highly semantically related to the source text. The extraction methods fail to predict these absent keyphrases. Therefore, generation methods have been proposed to produce a keyphrase verbatim from a predefined vocabulary, no matter whether the generated keyphrase appears in the source text. Compared with conventional extraction methods, generation methods have the ability of generating absent keyphrases as well as present keyphrases.

CopyRNN (Meng et al., 2017) is the first to employ the sequence-to-sequence (Seq2Seq) framework (Sutskever et al., 2014) with the copying mechanism (Gu et al., 2016) to generate keyphrases for the given documents. Following the CopyRNN, several Seq2Seq-based keyphrase generation approaches have been proposed to improve the generation performance (Chen et al., 2018; Ye and Wang, 2018; Chen et al., 2019; Zhao and Zhang, 2019; Wang et al., 2019; Yuan et al., 2020). All these existing methods generate present and absent keyphrases synchronously without ex-

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5717–5726
June 6–11, 2021. ©2021 Association for Computational Linguistics

Training(%)	Test(%)			
	Inspec	Krapivin	NUS	SemEval
49.79	13.12	11.74	11.30	11.25

Table 1: Proportions of absent keyphrases in training set and predictions of CopyRNN on four commonly used datasets, where top-10 predictions are considered.

plicitly distinguishing these two different categories of keyphrases, which leads to two problems: (1) They complicate the identification of present keyphrases. Specifically, they search for words over the entire predefined vocabulary containing a vast amount of words (*e.g.*, 50,000 words) to generate a present keyphrase verbatim, which is overparameterized since a present keyphrase can be simply selected from a continuous subsequence of the source text containing limited words (*e.g.*, less than 400 words). (2) They weaken the generation of absent keyphrases. Existing models for absent keyphrase generation are usually trained on datasets mixed with a large proportion of present keyphrases. Table 1 shows that nearly half of the training data are present keyphrases, which leads to the extremely low proportions of absent keyphrases generated by such a model, *i.e.*, CopyRNN. The above observation demonstrates that these methods are biased towards replicating words from source text for present keyphrase generation, which will inevitably affect the performance on generating absent keyphrases.

To address the aforementioned problems, we propose a **S**elect-**G**uide-**G**enerate (SGG) approach, which deals with present and absent keyphrase generation *separately* with different stages based on different mechanisms. Figure 1 illustrates an example of keyphrase prediction by SGG. The motivation behind is to solve keyphrase generation problem from selecting to generating, and use the selected results to guide the generation. Specifically, our SGG is implemented with a hierarchical neural network which performs Seq2Seq learning by applying a multi-task learning strategy. This network consists of a selector at low layer, a generator at high layer, and a guider at middle layer for information transfer. The selector generates present keyphrases through a pointing mechanism (Vinyals et al., 2015), which adopts attention distributions to select a sequence of words from the source text as output. The generator further generates the absent keyphrases through a pointing-generating (PG) mechanism (See et al., 2017). Since present keyphrases have already been generated by the selector, they should not be generated again by the generator. Therefore, a guider is designed to memorize the generated present keyphrases from the selector, and then fed into the attention module of the generator to constrain it to focus on generating absent keyphrases. We summarize our main contributions as follows:

- We propose a SGG approach which models *present* and *absent* keyphrase generation separately in different stages, *i.e.*, select, guide, and generate, without sacrificing the end-to-end training through back-propagation.
- Extensive experiments are conducted to verify the effectiveness of our model, which not only improves present keyphrase generation but also dramatically boosts the performance of absent keyphrase generation.
- Furthermore, we adopt SGG to a title generation task, and the experiment results indicate the extensibility and effectiveness of our SGG approach on generation tasks.

2 Related Work

As mentioned in Section 1, the extraction and generation methods are two different research directions in the field of keyphrase prediction. The existing extraction methods can be broadly classified into supervised and unsupervised approaches. The supervised approaches treat keyphrase extraction as a binary classification task, which train the models with the features of labeled keyphrases to determine whether a candidate phrase is a keyphrase (Witten et al., 1999; Medelyan et al., 2009; Gollapalli et al., 2017). In contrast, the unsupervised approaches treat keyphrase extraction as a ranking task, scoring each candidate using some different ranking metrics, such as clustering (Liu et al., 2009), or graph-based ranking (Mihalcea and Tarau, 2004; Wang et al., 2014; Gollapalli and Caragea, 2014; Zhang et al., 2017).

This work is mainly related to keyphrase generation approaches which have demonstrated good performance on keyphrase prediction task. Following CopyRNN (Meng et al., 2017), several extensions have been proposed to boost the generation capability. In CopyRNN, model training heavily relies on large amount of labeled data, which is often unavailable especially for the new domains. To address this problem, Ye and Wang (2018) proposed a semi-supervised keyphrase generation model that utilizes both abundant unlabeled data and limited labeled data. CopyRNN uses the concatenation of article title and abstract as input, ignoring the leading role of the title. To address this deficiency, Chen et al. (2019) proposed a title-guided Seq2Seq network to sufficiently utilize the already summarized information in title. In addition, some research attempts to introduce external knowledge into keyphrase generation, such as syntactic constraints (Zhao and Zhang, 2019) and latent topics (Wang et al., 2019).

These approaches do not consider the one-to-many relationship between the input text and target keyphrases, and thus fail to model the correlation among the multiple target keyphrases. To overcome this drawback, Chen et al. (2018) incorporated the review mechanism into keyphrase generation and proposed a model CorrRNN with correlation constraints. Similarly, SGG separately models one-to-many relationship between the input text and present keyphrases and absent keyphrases. To avoid generating duplicate keyphrases, Chen et al. (2020) proposed an exclusive hierarchical decoding framework that includes a hierarchical decoding process and either a soft or a hard exclusion mechanism. For the same purpose, our method deploys a guider to avoid the generator generating duplicate present keyphrases. Last but most important, all these methods do not consider the difference between present and absent keyphrases. We are the first to discriminately treat present and absent keyphrases in keyphrase generation task.

3 Methodology

3.1 Problem Definition

Given a dataset including K data samples, where the j-th data item $\langle x^{(j)}, y^{(j,p)}, y^{(j,a)} \rangle$ consists of a source text $x^{(j)}$, a set of present keyphrases $y^{(j,p)}$ and a set of absent keyphrases $y^{(j,a)}$. Different from CopyRNN (Meng et al., 2017) splitting each data item into multiple training examples, each of which contains only one keyphrase as target, we regard each data item as one training example by concatenating its present keyphrases as one target and absent keyphrases as another one. Specifically, assume that the j-th data item consists of m present keyphrases $\{y_1^{(j,p)}, ..., y_m^{(j,p)}\}$ and n absent keyphrases $\{y_1^{(j,a)}, ..., y_n^{(j,a)}\}$, the target present keyphrases $y^{(j,p)}$ and target absent keyphrases $y^{(j,a)}$ are represented as:

$$y^{(j,p)} = y_1^{(j,p)} \mid\mid y_2^{(j,p)} \mid\mid ... \mid\mid y_m^{(j,p)}$$

$$y^{(j,a)} = y_1^{(j,a)} \mid\mid y_2^{(j,a)} \mid\mid ... \mid\mid y_n^{(j,a)}$$

where $\mid\mid$ is a special splitter to separate the keyphrases. We then get the source text $x^{(j)}$, the present keyphrases $y^{(j,p)}$ and the absent keyphrases $y^{(j,a)}$ all as word sequences. Under this setting, our model is capable of generating multiple keyphrases in one sequence as well as capturing the mutual relations between these keyphrases. A keyphrase generation model is to learn the mapping from the source text $x^{(j)}$ to the target keyphrases $(y^{(j,p)}, y^{(j,a)})$. For simplicity, (x, y^p, y^a) is used to denote each item in the rest of this paper, where x denotes a source text sequence, y^p denotes its present keyphrase sequence and y^a denotes its absent keyphrase sequence.

3.2 Model Overview

The architecture of our proposed **S**elect-**G**uide-**G**enerate (SGG) approach is illustrated in Figure 2. Our model is the extension of Seq2Seq framework which consists of a **text encoder**, a **selector**, a **guider**, and a **generator**. The text encoder converts the source text x into a set of hidden representation vectors $\{\mathbf{h}_i\}_{i=1}^{L}$ with a bi-directional Long Short-term Memory Network (bi-LSTM) (Hochreiter and Schmidhuber, 1997), where L is the length of source text sequence. The selector is a uni-directional LSTM, which predicts the present keyphrase sequence y^p based on the attention distribution over source words. After selecting present keyphrases, a guider is produced by a guider to memorize the prediction information of the selector, and then fed to the attention module of a generator to adjust the information it pays attention to. The selection-guided generator is also implemented as a uni-directional LSTM, which produces the absent keyphrase sequence y^a based on two distributions over predefined-vocabulary and source words, respectively. At the same time, a soft switch gate p_{gen} is employed as a trade-off between the above two distributions.

3.3 Text Encoder

The goal of a text encoder is to provide a series of dense representations $\{\mathbf{h}_i\}_{i=1}^{L}$ of the source text. In our model, the text encoder is implemented as a bi-LSTM (Hochreiter and Schmidhuber, 1997) which reads an input sequence $x = \{x_i\}_{i=1}^{L}$ from

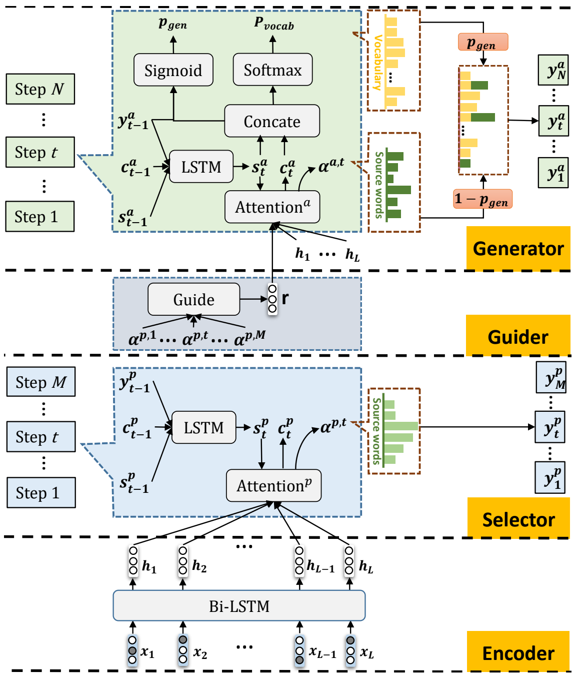


Figure 2: The architecture of the proposed SGG which is implemented with a hierarchical neural network.

two directions and outputs a sequence of forward hidden states $\{\overrightarrow{\mathbf{h}_i}\}_{i=1}^{L}$ and backward hidden states $\{\overleftarrow{\mathbf{h}_i}\}_{i=1}^{L}$ by iterating the following equations:

$$\overrightarrow{\mathbf{h}_i} = \texttt{LSTM}(x_i, \mathbf{h}_{i-1}) \quad (1)$$

$$\overleftarrow{\mathbf{h}_i} = \texttt{LSTM}(x_i, \mathbf{h}_{i+1}) \quad (2)$$

The final hidden representation $\mathbf{h}_i$ of the i-th source word is the concatenation of forward and backward hidden states, *i.e.*, $\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}]$.

3.4 Selector

A selector is designed to generate present keyphrase sequences through the pointer mechanism (Vinyals et al., 2015), which adopts the attention distribution as a pointer to select words from the source text as output. Specifically, given source text sequence x and previously generated words $\{y^p_1, ..., y^p_{t-1}\}$, the probability distribution of predicting next word y^p_t in present keyphrases is:

$$\mathcal{P}(y^p_t \mid y^p_{<t}, x) = \alpha^{p,t} = \texttt{softmax}(\mathbf{u}^{p,t}) \quad (3)$$

$$u^{p,t}_i = \mathbf{V}^T_p \texttt{tanh}(\mathbf{W}_p[\mathbf{s}^p_t; \mathbf{h}_i] + \mathbf{b}_p) \quad (4)$$

where $\alpha^{p,t}$ is the attention (Bahdanau et al., 2015) distribution at decoding time step t, $i \in (1, ..., L)$, and $\mathbf{V}_p$, $\mathbf{W}_p$ and $\mathbf{b}_p$ are trainable parameters of the model. $\mathbf{u}^{p,t}$ can be viewed as the degree of matching between input at position i and output at position t. $\mathbf{s}^p_t$ represents the hidden state at deciding time step t, and is updated by equation:

$$\mathbf{s}^p_t = \texttt{LSTM}(y^p_{t-1}, \mathbf{s}^p_{t-1}, \mathbf{c}^p_{t-1}) \quad (5)$$

where context vector $\mathbf{c}^p_{t-1} = \sum_{i=1}^{L} \alpha^{p,t-1}_i \mathbf{h}_i$ is the weighted sum of source hidden states.

3.5 Guider

A guider is designed to fully utilize the attention information of the selector to guide the generator on absent keyphrase generation. The idea behind is to utilize a guider $\mathbf{r}$ to softly indicate which words in source text have been generated by the selector. This is important for helping the generator to focus on generating the absent keyphrases. Specifically, $\mathbf{r}$ is constructed through the accumulation of the attention distributions over all decoding time steps of the selector, computed as:

$$\mathbf{r} = \sum_{t=1}^{M} \alpha^{p,t} \quad (6)$$

where M is the length of present keyphrase sequence. $\mathbf{r}$ is an unnormalized distribution over the source words. As the attention distribution of selector is equal to the probability distribution over the source words, $\mathbf{r}$ represents the possibility that these words have been generated by the selector. The calculation of guider is inspired by the coverage vector (Tu et al., 2016) that is sequentially updated during the decoding process. In contrast to this, the guider here is a static vector which is capable of memorizing a global information.

3.6 Selection-Guided Generator

A generator aims to predict an absent keyphrase sequence based on the guidance of the selection information from the guider. Unlike present keyphrases, most words in absent keyphrases do not appear in source text. Therefore, the generator generates absent keyphrases by picking up words from both a predefined large scale vocabulary and the source text (See et al., 2017; Gu et al., 2016). The probability distribution of predicting next word y^a_t in absent keyphrases is defined as:

$$\mathcal{P}(y^a_t \mid y^a_{<t}, x) = p_{gen}\mathcal{P}_{vocab}(y^a_t) + (1 - p_{gen}) \sum_{i:y^a_t = x_i} \alpha^{a,t}_i \quad (7)$$

where $\mathcal{P}_{vocab}$ is the probability distribution over the predefined vocabulary, which is zero if y^a_t is an out-of-vocabulary (OOV) word. Similarly, if y^a_t does not appear in the source text, then $\sum_{i:y_t^a=x_i} \alpha_i^{a,t}$ is zero. $\mathcal{P}_{vocab}$ is computed as:

$$\mathcal{P}_{vocab}(y_t^a) = \mathtt{softmax}(\mathbf{W}[\mathbf{s}_t^a; \mathbf{c}_t^a] + \mathbf{b}) \quad (8)$$

where $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters, $\mathbf{s}_t^a$ is the hidden state of generator, and $\mathbf{c}_t^a$ is the context vector for generating absent keyphrase sequence, computed by the following equations:

$$\mathbf{c}_t^a = \sum_{i=1}^{L} \alpha_i^{a,t} \mathbf{h}_i \quad (9)$$

$$\alpha^{a,t} = \mathtt{softmax}(\mathbf{u}^{a,t}) \quad (10)$$

$$u_i^{a,t} = \mathbf{V}_a^T \mathtt{tanh}(\mathbf{W}_a[\mathbf{s}_t^a; \mathbf{h}_i; \mathbf{r}] + \mathbf{b}_a) \quad (11)$$

where $\mathbf{V}_a$, $\mathbf{W}_a$ and $\mathbf{b}_a$ are learnable parameters. $\mathbf{r}$ is a vector produced by the guider. The generation probability p_{gen} at time step t is computed as:

$$p_{gen} = \sigma(\mathbf{W}_{gen}[\mathbf{c}_t^a; \mathbf{s}_t^a; \mathtt{emb}(y_{t-1}^a)] + \mathbf{b}_{gen}) \quad (12)$$

where $\mathbf{W}_{gen}$ and $\mathbf{b}_{gen}$ are learnable parameters, $\sigma(\cdot)$ represents a sigmoid function and $\mathtt{emb}(y_{t-1}^a)$ is the embedding of y_{t-1}^a. In addition, p_{gen} in formula (7) is used as a soft switch to choose either generating words over vocabulary or copying words from source text based on distribution $\alpha^{a,t}$.

3.7 Training

Given the set of data pairs $\{x^{(j)}, y^{(j,p)}, y^{(j,a)}\}_{j=1}^{K}$, the loss function of the keyphrase generation consists of two parts of cross entropy losses:

$$\mathcal{L}_p(\theta) = -\sum_{j=1}^{K}\sum_{i=1}^{M} log(\mathcal{P}(y_i^{(j,p)}|x^{(j)};\theta)) \quad (13)$$

$$\mathcal{L}_a(\theta) = -\sum_{j=1}^{K}\sum_{i=1}^{N} log(\mathcal{P}(y_i^{(j,a)}|x^{(j)};\theta)) \quad (14)$$

where $\mathcal{L}_p$ and $\mathcal{L}_a$ are the losses of generating present and absent keyphrases, respectively. N is the word sequence length of absent keyphrases, and θ are the parameters in our model. The training objective is to jointly minimize the two losses:

$$\mathcal{L} = \mathcal{L}_p + \mathcal{L}_a. \quad (15)$$

4 Experiment

4.1 Dataset

We use the dataset collected by Meng et al. (2017) from various online digital libraries, which contains approximately 570K samples, each of which contains a title and an abstract of a scientific publication as source text, and author-assigned keywords as target keyphrases. We randomly select the example which contains at least one present keyphrase to construct the training set. Then, a validation set containing 500 samples will be selected from the remaining examples. In order to evaluate our proposed model comprehensively, we test models on four widely used public datasets from the scientific domain, namely Inspec (Hulth and Megyesi, 2006), Krapivin (Krapivin et al., 2009), SemEval-2010 (Kim et al., 2010) and NUS (Nguyen and Kan, 2007), the statistic information of which are summarized in Table 2.

Dataset		**#Abs**	**#PKPs**	**#AKPs**
Test	Inspec	500	3,654	1,349
	Krapivin	400	1,299	1,040
	NUS	211	1,333	1,128
	SemEval	100	625	841
Validation		500	1,158	1,418
Training		453,757	1,082,285	1,073,404

Table 2: Statistics of the dataset. #Abs, #PKPs, #AKPs denote the number of abstracts, present keyphrases, and absent keyphrases, respectively.

4.2 Baselines and Evaluation Metrics

For present keyphrase prediction, we compare our model with both extraction and generation approaches. Extraction approaches include two unsupervised extraction methods: TF-IDF, TextRank (Mihalcea and Tarau, 2004) and one classic supervised extraction method KEA (Witten et al., 1999). For the generation baselines, some models, such as CopyRNN, split each data item into multiple training examples, each of which only contains one keyphrase, while the other models concatenate all keyphrases as target. To simplicity, the pattern of training model only with one keyphrase is denoted as **one-to-one** and with the concatenation of all keyphrases as **one-to-many**. The generation baselines are the following state-of-the-art encoder-decoder models:

- **CopyRNN(one-to-one)** (Meng et al., 2017) represents a RNN-based encoder-decoder model incorporating the copying mechanism.

- **CopyTrans(one-to-many)** is a transformer-based (Vaswani et al., 2017) encoder-decoder model incorporating the copying mechanism.

Method	Inspec		Krapivin		NUS		SemEval	
	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10
TF-IDF	22.1	31.3	12.9	16.0	13.6	18.4	12.8	19.4
TextRank	22.3	28.1	18.9	16.2	19.5	19.6	17.6	18.7
KEA	9.8	12.6	11.0	15.2	6.9	8.4	2.5	2.6
CopyRNN	27.8	34.2	31.1	26.6	33.4	32.6	29.3	30.4
CopyTrans†	21.1	16.2	26.4	20.5	35.1	28.2	29.5	26.3
CorrRNN	–	–	**31.8**	**27.8**	35.8	33.0	32.0	32.0
CatSeq	29.0	30.0	30.7	27.4	35.9	34.9	30.2	30.6
SGG	**30.6**	**35.9**	28.8	25.3	**36.3**	**35.8**	**33.8**	**33.6**

Table 3: F1@5/10 results of predicting present keyphrases of different models on four datasets. The best and second best performance in each column are highlighted with bold and underline respectively. † indicates that the model is reimplemented.

Method	Inspec	Krapivin	NUS	SemEval
CopyRNN	10.0	20.2	11.6	**6.7**
CopyTrans†	5.6	16.9	8.9	4.1
CorrRNN†	8.5	15.2	8.0	3.5
CatSeq	2.9	7.4	3.1	2.5
SGG	**11.0**	**23.5**	**12.4**	4.9

Table 4: Recall@50 results of predicting absent keyphrases of different models on four datasets. The CorrRNN is retrained following the implementation details in Chen et al. (2018) as they did not report the Recall@50 results.

- **CorrRNN(one-to-many)** (Chen et al., 2018) is an extension of CopyRNN incorporating the coverage mechanism (Tu et al., 2016).
- **CatSeq(one-to-many)** (Yuan et al., 2020) has the same model structure as CopyRNN. The difference is CatSeq is trained by one-to-many.

The baseline CopyTrans has not been reported in existing papers and thus is retrained. The implementation of Transformer is base on open source tool OpenNMT [1]. For our experiments of absent keyphrase generation, only generation methods are chosen as baselines. The copying mechanism used in all reimplemented generation models is based on the version (See et al., 2017), which is slightly different from the implementations by version (Meng et al., 2017; Gu et al., 2016). **SGG** indicates the full version of our proposed model, which contains a selector, a guider, and a generator. Note that SGG is also trained under one-to-many pattern.

Same as CopyRNN, we adopt top-N macro-averaged *F-measure* (F1) and *recall* as our evaluation metrics for the present and absent keyphrases respectively. The choice of larger N (*i.e.*, 50 v.s. 5 and 10) for absent keyphrase is due to the fact that absent keyphrases are more difficult to be generated than present keyphrases. For present keyphrase evaluation, exact match is used for determining whether the predictions are correct. For absent keyphrase evaluation, Porter Stemmer is used to stem all the words in order to remove words' suffix before comparisons.

4.3 Implementation Details

We set maximal length of source sequence as 400, 25 for target sequence of selector and generator, and 50 for the decoders of all generation baselines. We choose the top 50,000 frequently-occurred words as our vocabulary. The dimension of the word embedding is 128. The dimension of hidden state in encoder, selector and generator is 512. The word embedding is randomly initialized and learned during training. We initialize the parameters of models with uniform distribution in [-0.2,0.2]. The model is optimized using Adagrad (Duchi et al., 2011) with learning rate = 0.15, initial accumulator = 0.1 and maximal gradient normalization = 2. In the inference process, we use beam search to generate diverse keyphrases and the beam size is 200 same as baselines. All the models are trained on a single Tesla P40.

4.4 Results and Analysis

In this section, we present the results of present and absent keyphrase generation separately. The results of predicting present keyphrases are shown in Table 3, in which the F1 at top-5 and top-10 predictions are given. We first compare our proposed

[1] https://github.com/OpenNMT/OpenNMT-py

Method	Inspec	Krapivin	NUS	SemEval
CopyRNN	13.12	11.74	11.30	11.25
SGG	79.16	79.28	76.02	79.20

Table 5: Proportion of absent keyphrases in the predictions of CopyRNN and generator. The proportion of CopyRNN is same as Table 1.

model with the conventional keyphrase extraction methods. The results show that our model performs better than extraction methods with a large margin, demonstrating the potential of the Seq2Seq-based generation models in automatic keyphrase extraction task. We then compare our model with the generation baselines, and the results indicate that our model still outperforms these baselines significantly. The better performance of SGG illustrates the pointing based selector is sufficient and more effective to generate present keyphrase.

We further analyze the experimental results of absent keyphrase generation. Table 4 presents the recall results of the generation baselines and our model on four datasets. It can be observed that our model significantly improves the performance of absent keyphrase generation, compared to the generation baselines. This is because SGG is equipped with a generator that is not biased to generate present keyphrases and the designed guider in SGG further guides the generator to focus on generating absent keyphrases. Table 5 shows the proportion of absent keyphrases generated by SGG. The comparison of Table 1 and 5 demonstrates that our model have the ability to generate large portions of absent keyphrases rather than tending to generate present keyphrases.

In addition, an interesting phenomenon can be found from the results of CopyRNN and CatSeq that one-to-one pattern generally performs better than one-to-many if under the same model structure in absent keyphrase generation. To explore this phenomenon, we use the same code, same training set to retrain CopyRNN under one-to-one and one-to-many patterns, and the test results show that one-to-one could boost the performance in absent keyphrase generation. However, SGG cannot be trained under one-to-one pattern as the core of guider in SGG is to memory all present keyphrases. Even so, SGG still has better performance than CopyRNN. The results of SGG achieve 1.6% average gain than CopyRNN and 31.8% average gain than the best-performing results of one-to-many baselines over four test sets.

4.5 SGG for Title Generation

In this section, we explore the extensibility of SGG in other natural language generation (NLG) tasks, *i.e.*, title generation. We adopt the same dataset described in Section 4.1 for title generation, which contains abstracts, present keyphrases, absent keyphrases, and titles. Specifically, a title generation model takes an abstract as input and generates a title as output. To train SGG model for title generation, present keyphrases appearing in the titles are used as labels to train the selectors[2], and the titles are used to train the generators. The idea behind is to utilize the present keyphrase generation as an auxiliary task to help the main title generation task. In order to evaluate SGG on title generation, we choose models CopyTrans and pointer-generator (PG-Net) (See et al., 2017) as baselines. We use ROUGE-1 (unigram), ROUGE-2 (bi-gram), ROUGE-L (LCS) and human evaluation as evaluation metrics. For human evaluation, we randomly selects 100 abstracts for each test set, then distribute them to four people on average. The evaluation standard is the fluency of generated title and whether it correctly provides the core topics of an abstract.

Inspec	RG-1	RG-2	RG-L	Human
CopyTrans	83.58	43.81	45.25	74/100
PG-Net	83.03	43.44	45.20	77/100
SGG	**84.25**	**44.98**	**46.87**	**83/10**0
Krapivin	**RG-1**	**RG-2**	**RG-L**	**Human**
CopyTrans	84.23	50.01	50.63	89/100
PG-Net	84.75	50.82	51.48	87/100
SGG	**84.96**	**51.35**	**52.34**	**90/100**
NUS	**RG-1**	**RG-2**	**RG-L**	**Human**
CopyTrans	86.76	**54.90**	52.49	82/100
PG-Net	86.59	52.59	50.61	79/100
SGG	**87.01**	**54.90**	**52.57**	**89/100**
SemEval	**RG-1**	**RG-2**	**RG-L**	**Human**
CopyTrans	86.92	**55.10**	53.05	82/100
PG-Net	86.68	50.16	51.31	78/100
SGG	**87.54**	53.38	**53.55**	**84/100**

Table 6: Results of title generation of various models on four datasets.

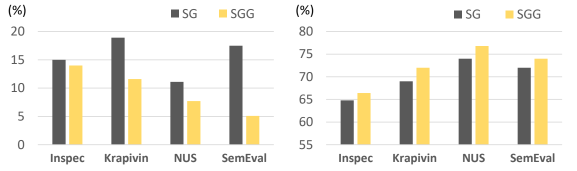
The results of title generation are shown in Table 6, from which we observe that our proposed

[2] The present keyphrase information used for training SGG is not used during inference. Datasets without given present keyphrases should consider to conduct labeling.

Dataset	Absent keyphrase generation	Title generation			
	Recall@50	RG-1	RG-2	RG-L	BLEU-4
Inspec	8.6(-2.4)	83.51(-0.74)	44.40(-0.58)	45.80(-1.07)	11.02(+0.41)
Krapivin	23.2(-0.3)	84.56(-0.40)	50.56(-0.79)	50.87(-0.48)	11.46(-1.38)

Table 7: Results of SG on absent keyphrase generation and title generation tasks. (±) indicates the comparison of the results of SG and SGG. The results of SGG please refer to Table 4 and Table 6.

model SGG achieves better performance than the strong baselines on all datasets, proving that SGG could be directly applied to title generation task and still keep highly effective.



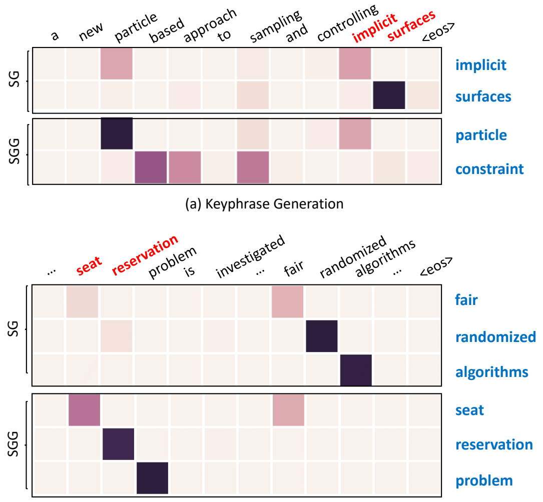
(a) Keyphrase Generation (b) Title Generation

Figure 3: Proportions of test examples that the predictions of generator overlap with the predictions of selector. Here only the top-1 predictions of generator and selector are used.

4.6 Ablation Study on Guider

In this section, we further study the effectiveness of our proposed guider module. Table 7 displays the results of SG (only a **s**elector, a **g**enerator, no guider) and its comparison with SGG on the two largest test sets Inspec and Krapivin, which illustrates that the guider has a remarkable effect on absent keyphrase and title generation tasks.

In more detail, we analyze that the function of guiders on these two tasks is different, which depends on the correlation between the targets of selector and generator. For example, in the task of keyphrase generation, the words predicted from selector should not be repeatedly generated by generator because the present keyphrases and absent keyphrases in a given text usually do not have overlapping words. However, in the task of title generation, the selected words by selector should be paid more attention on by generator since they are usually part of the target titles. To verify the above analysis, we visualize two examples of the attention scores in generators for the two tasks in Figure 4. For keyphrase generation, SG repeatedly generates "implicit surfaces" that has already been generated by its selector. In contrast, SGG successfully avoids this situation and it correctly generates the absent keyphrase "particle constraint". For title generation, the guider helps SGG to assign higher attention scores to the words in "seat reservation" that has been generated by selector.



(a) Keyphrase Generation

(b) Title Generation

Figure 4: Visualization of attention score in generator for keyphrase generation and title generation. The words marked in red have already been generated by the selector. The words marked in blue are the generation of the generator. In these two examples, phrase "particle constraint" is the correct absent keyphrase for keyphrase generation and "seat reservation problem" is part of the correct title for title generation.

Figure 3 gives the proportion of test examples that the predictions of generator overlap with the predictions of selector. We observe that SG is more likely to generate the words that have been generated by selector than SGG in keyphrase generation. In contrast, the results on title generation indicate that SGG is more likely to generate previously selected words than SG for this task. Through the analysis above, we conjecture that the guider is able to correctly guide the behaviour of generator in different tasks, *i.e.*, learn to encourage or discourage generating previously selected words.

5 Conclusion

In this paper, a Select-Guide-Generate (SGG) approach is proposed and implemented with a hierarchical neural model for keyphrase generation, which separately deals with the generation of present and absent keyphrases. Comprehensive empirical studies demonstrate the effectiveness of SGG. Furthermore, a title generation task indicates the extensibility of SGG in other generation tasks.

6 Acknowledgments

This work is supported by the National Key Research and Development Program of China under Grant No. 2018YFB2100802.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of ICLR*.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Proceedings of EMNLP*.

Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive hierarchical decoding for deep keyphrase generation. *arXiv preprint arXiv:2004.08511*.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019. Title-guided encoding for keyphrase generation. In *Proceedings of AAAI*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of AAAI*.

Sujatha Das Gollapalli, Xiao-Li Li, and Peng Yang. 2017. Incorporating expert knowledge into keyphrase extraction. In *Proceedings of AAAI*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Anette Hulth and Beáta B Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of ACL*.

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*.

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

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of EMNLP*.

Olena Medelyan, Eibe Frank, and Ian H Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of EMNLP*.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of ACL*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of EMNLP*.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of 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 ACL*.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural models for key phrase extraction and question generation. In *Proceedings of ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of NIPS*.

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

Fang Wang, Zhongyuan Wang, Senzhang Wang, and Zhoujun Li. 2014. Exploiting description knowledge for keyphrase extraction. In *Proceedings of PRICAI*.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of ACL*.

Yue Wang, Jing Li, Hou Pong Chan, Irwin King, Michael R. Lyu, and Shuming Shi. 2019. Topic-aware neural keyphrase generation for social media language. In *Proceedings of ACL*.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevillmanning. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of ACM Conference on Digital Libraries*.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of EMNLP*.

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 ACL*.

Yuxiang Zhang, Yaocheng Chang, Xiaoqing Liu, Sujatha Das Gollapalli, Xiaoli Li, and Chunjing Xiao. 2017. Mike: keyphrase extraction by integrating multidimensional information. In *Proceedings of CIKM*.

Jing Zhao and Yuxiang Zhang. 2019. Incorporating linguistic constraints into keyphrase generation. In *Proceedings of ACL*.