

# Title-Guided Encoding for Keyphrase Generation

Wang Chen,<sup>1,2</sup> Yifan Gao,<sup>1,2</sup> Jiani Zhang,<sup>1,2</sup> Irwin King,<sup>1,2</sup> Michael R. Lyu<sup>1,2</sup>

<sup>1</sup>Department of Computer Science and Engineering,

The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

<sup>2</sup>Shenzhen Key Laboratory of Rich Media Big Data Analytics and Application,

Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China

{wchen, yfgao, jnzhang, king, lyu}@cse.cuhk.edu.hk

## Abstract

Keyphrase generation (KG) aims to generate a set of keyphrases given a document, which is a fundamental task in natural language processing (NLP). Most previous methods solve this problem in an extractive manner, while recently, several attempts are made under the generative setting using deep neural networks. However, the state-of-the-art generative methods simply treat the document title and the document main body equally, ignoring the leading role of the title to the overall document. To solve this problem, we introduce a new model called Title-Guided Network (TG-Net) for automatic keyphrase generation task based on the encoder-decoder architecture with two new features: (i) the title is additionally employed as a query-like input, and (ii) a title-guided encoder gathers the relevant information from the title to each word in the document. Experiments on a range of KG datasets demonstrate that our model outperforms the state-of-the-art models with a large margin, especially for documents with either very low or very high title length ratios.

## Introduction

Keyphrases are short phrases that can quickly provide the main information of a given document (the terms “document”, “source text” and “context” are interchangeable in this study, and all of them represent the concatenation of the title and the main body.). Because of the succinct and accurate expression, keyphrases are widely used in information retrieval (Jones and Staveley 1999), document categorizing (Hulth and Megyesi 2006), opinion mining (Berend 2011), etc. Due to the huge potential value, various automatic keyphrase extraction and generation methods have been developed. As shown in Figure 1, the input usually consists of the title and the main body, and the output is a set of keyphrases.

Most typical automatic keyphrase extraction methods (Witten et al. 1999; Medelyan, Frank, and Witten 2009; Mihalcea and Tarau 2004) focus on extracting **present keyphrases** like “relevance profiling” in Figure 1, which are the exact phrases appearing in the source text. The main ideas among them are *identifying candidate phrases first and then ranking* algorithms. However, these methods ignore the

**Title:** *Within-document retrieval*: A user-centred evaluation of *relevance profiling*

**Main Body:** We present a user-centred, task-oriented, comparative evaluation of two *within-document retrieval* tools. ProfileSkim computes a relevance profile for a document with respect to a query, and presents the profile as an interactive bar graph. ... *Relevance profiling* should prove highly beneficial for users trying to identify relevant information within long documents. ...

- (a) Present Keyphrases:  
{within-document retrieval; relevance profiling}
- (b) Absent Keyphrases:  
{interactive information retrieval; task-oriented evaluation; language models}

Figure 1: An example of keyphrase generation. The present keyphrases are bold and italic in the source text.

semantic meaning underlying the context content and cannot generate **absent keyphrases** like “interactive information retrieval”, which do not appear in the source text.

To overcome the above drawbacks, several encoder-decoder based keyphrase generation methods have been proposed including CopyRNN (Meng et al. 2017) and CopyCNN (Zhang, Fang, and Weidong 2017). First, these methods treat the title and the main body equally and concatenate them as the only source text input. Then, the encoder maps each source text word into a hidden state vector which is regarded as the contextual representation. Finally, based on these representations, the decoder generates keyphrases from a predefined vocabulary regardless of the presence or absence of the keyphrases. A serious drawback of these models is that they ignore the leading role of the title and consequently fail to sufficiently utilize the already summarized information in it.

It is a widely agreed fact that the title can be viewed as a high-level summary of a document and the keyphrases provide more details of the key topics introduced in the document (Li et al. 2010). They play a similar and complementary role with each other. Therefore, keyphrases should have close semantic meaning to the title (Li et al. 2010). For example, as shown in Figure 1, the title contains most of the salient points reflected by these keyphrases including “re-

trieval”, “profiling”, and “evaluation”. Statistically, we study the proportion of keyphrases related to the title on the largest KG dataset and show the results in Table 1. For simplicity, we define a *TitleRelated* keyphrase as the keyphrase containing at least one common non-stop-word with the title. From Table 1, we find that about 33% absent keyphrases are *TitleRelated*. For present keyphrases, the *TitleRelated* percentage is up to around 60%. By considering the fact that the length of a title is usually only 3%-6% of the corresponding source text, we can conclude that the title, indeed, contains highly summative and valuable information for generating keyphrases. Moreover, information in the title is also helpful in reflecting which part of the main body is essential, such as the part containing the same or related information with the title. For instance, in Figure 1, the point “evaluation” in the title can guide us to focus on the part “... task-oriented, comparative evaluation ...” of the main body, which is highly related to the absent keyphrase “task-oriented evaluation”.

To sufficiently leverage the title content, we introduce a new title-guided network by taking the above fact into the keyphrase generation scenario. In our model, the title is additionally treated as a query-like input in the encoding stage. First, two bi-directional Gated Recurrent Unit (GRU) (Cho et al. 2014) layers are adopted to separately encode the context and the title into corresponding contextual representations. Then, an attention-based matching layer is used to gather the relevant title information for each context word according to the semantic relatedness. Since the context is the concatenation of the title and the main body, this layer implicitly contains two parts. The former part is the “title to title” self-matching, which aims to make the salient information in the title more important. The latter part is the “main body to title” matching wherein the title information is employed to reflect the importance of information in the main body. Next, an extra bi-directional GRU layer is used to merge the original contextual information and the corresponding gathered title information into the final title-guided representation for each context word. Finally, the decoder equipped with attention and copy mechanisms utilizes the final title-guided context representation to predict keyphrases.

We evaluate our model on five real-world benchmarks, which test the ability of our model to predict present and absent keyphrases. Using these benchmarks, we demonstrate that our model can effectively exploit the title information and it outperforms the relevant baselines by a significant margin: for present (absent) keyphrase prediction, the improvement gain of F1-measure at 10 (Recall at 50) score is up to 9.4% (19.1%) compared to the best baseline on the largest dataset. Besides, we probe the performance of our model and a strong baseline CopyRNN on documents with different title length ratios (i.e., the title length over the context length). Experimental results show that our model consistently improves the performance with large gains, especially for documents with either very low or very high title length ratios.

Our main contributions consist of three parts:

- A new perspective on keyphrase generation is explored, which sufficiently employs the title to guide the keyphrase prediction process.

|         | Keyphrase | TitleRelated | %     |
|---------|-----------|--------------|-------|
| Present | 54,403    | 32,328       | 59.42 |
| Absent  | 42,997    | 14,296       | 33.25 |

Table 1: The statistics of *TitleRelated* keyphrases on the validation set of **KP20k**.

- A novel TG-Net model is proposed, which can effectively leverage the useful information in the title.
- The overall empirical results on five real-world benchmarks show that our model outperforms the state-of-the-art models significantly on both present and absent keyphrase prediction, especially for documents with either very low or very high title length ratios.

## Related Work

### Automatic Keyphrase Extraction

Most of the automatic keyphrase extraction methods consist of two steps. Firstly, the *candidate identification* step obtains a set of candidate phrases such as phrases with some specific part-of-speech (POS) tags (Medelyan, Frank, and Witten 2009; Witten et al. 1999). Secondly, in the *ranking* step, all the candidates are ranked based on the importance computed by either unsupervised ranking approaches (Wan and Xiao 2008; Mihalcea and Tarau 2004; Florescu and Caragea 2017) or supervised machine learning approaches (Medelyan, Frank, and Witten 2009; Witten et al. 1999; Nguyen and Luong 2010; Florescu and Jin 2018). Finally, the top-ranked candidates are selected as the keyphrases. Besides these widely developed two-step approaches, there are also some methods using a sequence labeling operation to extract keyphrases (Zhang et al. 2016; Luan, Ostendorf, and Hajishirzi 2017; Gollapalli, Li, and Yang 2017). But they still cannot generate absent keyphrases.

Some extraction approaches (Li et al. 2010; Liu et al. 2011) also consider the influence of the title. Li et al. (2010) proposes a graph-based ranking algorithm which initializes the importance score of title phrases as one and the others as zero and then propagates the influence of title phrases iteratively. The biggest difference between Li et al. (2010) and our method is that our method utilizes the contextual information of the title to guide the context encoding, while their model only considers the phrase occurrence in the title. Liu et al. (2011) models keyphrase extraction process as a translation operation from a document to keyphrases. The title is used as the target output to train the translator. Compared with our model, one difference is that this method still cannot handle semantic meaning of the context. The other is that our model regards the title as an extra query-like input instead of a target output.

### Automatic Keyphrase Generation

Keyphrase generation is an extension of keyphrase extraction which explicitly considers the absent keyphrase prediction. CopyRNN (Meng et al. 2017) first frames the generation process as a sequence-to-sequence learning

task and employs a widely used encoder-decoder framework (Sutskever, Vinyals, and Le 2014) with attention (Bahdanau, Cho, and Bengio 2015) and copy (Gu et al. 2016) mechanisms. Based on CopyRNN, various extensions (Hai and Lu 2018; Jun et al. 2018) are recently proposed. However, these recurrent neural network (RNN) based models may suffer the low-efficiency issues because of the computation dependency between the current time step and the preceding time steps in RNN. To overcome this shortcoming, CopyCNN (Zhang, Fang, and Weidong 2017) applies a convolutional neural network (CNN) based encoder-decoder model (Gehring et al. 2017). CopyCNN employs position embedding for obtaining a sense of order in the input sequence and adopts gated linear units (GLU) (Dauphin et al. 2017) as the non-linearity function. CopyCNN not only achieves much faster keyphrase generation speed but also outperforms CopyRNN on five real-world benchmark datasets.

Nevertheless, both CopyRNN and CopyCNN treat the title and the main body equally, which ignores the semantic similarity between the title and the keyphrases. Motivated by the success of query-based encoding in various natural language processing tasks (Gao et al. 2018; Song, Wang, and Hamza 2017; Nema et al. 2017; Wang et al. 2017), we regard the title as an extra query-like input to guide the source context encoding. Consequently, we propose a TG-Net model to explicitly explore the useful information in the title. In this paper, we focus on how to incorporate a title-guided encoding into the RNN-based model, but it is also convenient to apply this idea to the CNN-based model in a similar way.

## Problem Definition

We denote vectors with bold lowercase letters, matrices with bold uppercase letters and sets with calligraphy letters. We denote  $\Theta$  as a set of parameters and  $\mathbf{W}$  as a parameter matrix.

Keyphrase generation (KG) is usually formulated as follows: given a context  $\mathbf{x}$ , which is the concatenation of the title and the main body, output a set of keyphrases  $\mathcal{Y} = \{\mathbf{y}^i\}_{i=1,\dots,M}$  where  $M$  is the keyphrase number of  $\mathbf{x}$ . Here, the context  $\mathbf{x} = [x_1, \dots, x_{L_x}]$  and each keyphrase  $\mathbf{y}^i = [y_1^i, \dots, y_{L_{y^i}}^i]$  are both word sequences, where  $L_x$  is the length (i.e., the total word number) of the context and  $L_{y^i}$  is the length of the  $i$ -th produced keyphrase  $\mathbf{y}^i$ .

To adapt the encoder-decoder framework,  $M$  context-keyphrase pairs  $\{(\mathbf{x}, \mathbf{y}^i)\}_{i=1,\dots,M}$  are usually split. Since we additionally use the title  $\mathbf{t} = [t_1, \dots, t_{L_t}]$  with length  $L_t$  as an extra query-like input, we split  $M$  context-title-keyphrase triplets  $\{(\mathbf{x}, \mathbf{t}, \mathbf{y}^i)\}_{i=1,\dots,M}$  instead of context-keyphrase pairs to feed our model. For conciseness, we use  $(\mathbf{x}, \mathbf{t}, \mathbf{y})$  to represent such a triplet, where  $\mathbf{y}$  is one of its target keyphrases.

## Our Proposed Model

### Title-Guided Encoder Module

As shown in Figure 2, the title-guided encoder module consists of a sequence encoding layer, a matching layer, and a

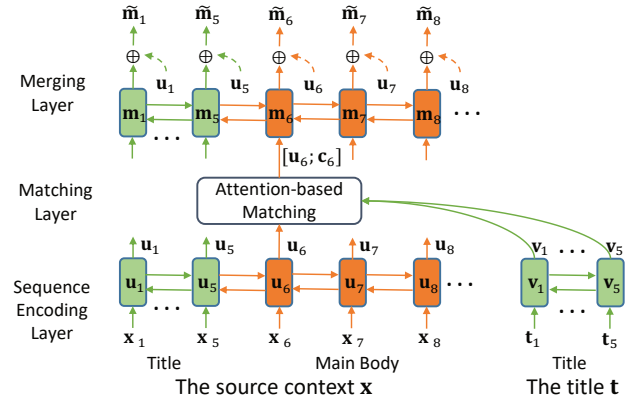


Figure 2: The title-guided encoder module. (Best viewed in color.)

merging layer. First, the sequence encoding layer reads the context input and the title input and learns their contextual representations separately. Then the matching layer gathers the relevant title information for each context word reflecting the important parts of the context. Finally, the merging layer merges the aggregated title information into each context word producing the final title-guided context representation.

**Sequence Encoding Layer** At first, an embedding lookup table is applied to map each word within the context and the title into a dense vector with a fixed size  $d_e$ . To incorporate the contextual information into the representation of each word, two bi-directional GRUs (Cho et al. 2014) are used to encode the context and the title respectively:

$$\vec{\mathbf{u}}_i = \text{GRU}_{11}(\mathbf{x}_i, \vec{\mathbf{u}}_{i-1}), \quad (1)$$

$$\overleftarrow{\mathbf{u}}_i = \text{GRU}_{12}(\mathbf{x}_i, \overleftarrow{\mathbf{u}}_{i+1}), \quad (2)$$

$$\vec{\mathbf{v}}_j = \text{GRU}_{21}(\mathbf{t}_j, \vec{\mathbf{v}}_{j-1}), \quad (3)$$

$$\overleftarrow{\mathbf{v}}_j = \text{GRU}_{22}(\mathbf{t}_j, \overleftarrow{\mathbf{v}}_{j+1}), \quad (4)$$

where  $i = 1, 2, \dots, L_x$  and  $j = 1, 2, \dots, L_t$ .  $\mathbf{x}_i$  and  $\mathbf{t}_j$  are the  $d_e$ -dimensional embedding vectors of the  $i$ -th context word and the  $j$ -th title word separately.  $\vec{\mathbf{u}}_i$ ,  $\overleftarrow{\mathbf{u}}_i$ ,  $\vec{\mathbf{v}}_j$ , and  $\overleftarrow{\mathbf{v}}_j$  are  $d/2$ -dimensional hidden vectors where  $d$  is the hidden dimension of the bi-directional GRUs. The concatenations  $\mathbf{u}_i = [\vec{\mathbf{u}}_i; \overleftarrow{\mathbf{u}}_i] \in \mathbb{R}^d$  and  $\mathbf{v}_j = [\vec{\mathbf{v}}_j; \overleftarrow{\mathbf{v}}_j] \in \mathbb{R}^d$  are used as the contextual vectors for the  $i$ -th context word and the  $j$ -th title word respectively.

**Matching Layer** The attention-based matching layer is engaged to aggregate the relevant information from the title for each word within the context. The aggregation operation  $\mathbf{c}_i = \text{attn}(\mathbf{u}_i, [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{L_t}]; \mathbf{W}_1)$  is as follows:

$$\mathbf{c}_i = \sum_{j=1}^{L_t} \alpha_{i,j} \mathbf{v}_j, \quad (5)$$

$$\alpha_{i,j} = \exp(s_{i,j}) / \sum_{k=1}^{L_t} \exp(s_{i,k}), \quad (6)$$

$$s_{i,j} = (\mathbf{u}_i)^T \mathbf{W}_1 \mathbf{v}_j, \quad (7)$$

where  $\mathbf{c}_i \in \mathbb{R}^d$  is the aggregated information vector for the  $i$ -th word of  $\mathbf{x}$ .  $\alpha_{i,j}$  ( $s_{i,j}$ ) is the normalized (unnormalized) attention score between  $\mathbf{u}_i$  and  $\mathbf{v}_j$ .

Here, the matching layer is implicitly composed of two parts because the context is a concatenation of the title and the main body. The first part is the ‘‘title to title’’ self-matching part, wherein each title word attends the whole title itself and gathers the relevant title information. This part is used to strengthen the important information in the title itself, which is essential to capture the core information because the title already contains much highly summative information. The other part is the ‘‘main body to title’’ matching part wherein each main body word also aggregates the relevant title information based on semantic relatedness. In this part, the title information is employed to reflect the importance of information in the main body based on the fact that the highly title-related information in the main body should contain core information. Through these two parts, this matching layer can utilize the title information much more sufficiently than any of the previous sequence to sequence methods.

**Merging Layer** Finally, the original contextual vector  $\mathbf{u}_i$  and the aggregated information vector  $\mathbf{c}_i$  are used as the inputs to another information merging layer:

$$\vec{\mathbf{m}}_i = \text{GRU}_{31}([\mathbf{u}_i; \mathbf{c}_i], \vec{\mathbf{m}}_{i-1}), \quad (8)$$

$$\overleftarrow{\mathbf{m}}_i = \text{GRU}_{32}([\mathbf{u}_i; \mathbf{c}_i], \overleftarrow{\mathbf{m}}_{i+1}), \quad (9)$$

$$\tilde{\mathbf{m}}_i = \lambda \mathbf{u}_i + (1 - \lambda)[\vec{\mathbf{m}}_i; \overleftarrow{\mathbf{m}}_i], \quad (10)$$

where  $[\mathbf{u}_i; \mathbf{c}_i] \in \mathbb{R}^{2d}$ ,  $\vec{\mathbf{m}}_i \in \mathbb{R}^{d/2}$ ,  $\overleftarrow{\mathbf{m}}_i \in \mathbb{R}^{d/2}$ ,  $[\vec{\mathbf{m}}_i; \overleftarrow{\mathbf{m}}_i] \in \mathbb{R}^d$ , and  $\tilde{\mathbf{m}}_i \in \mathbb{R}^d$ . The  $\mathbf{u}_i$  in Eq. (10) is a residual connection, and  $\lambda \in (0, 1)$  is the corresponding hyperparameter. Eventually, we obtain the title-guided contextual representation of the context (i.e.,  $[\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \dots, \tilde{\mathbf{m}}_{L_x}]$ ), which is regarded as a memory bank for the later decoding process.

## Decoder Module

After encoding the context into the title-guided representation, we engage an attention-based decoder (Luong, Pham, and Manning 2015) incorporating with copy mechanism (See, Liu, and Manning 2017) to produce keyphrases. Only one forward GRU is used in this module:

$$\mathbf{h}_t = \text{GRU}_4([\mathbf{e}_{t-1}; \tilde{\mathbf{h}}_{t-1}], \mathbf{h}_{t-1}), \quad (11)$$

$$\hat{\mathbf{c}}_t = \text{attn}(\mathbf{h}_t, [\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \dots, \tilde{\mathbf{m}}_{L_x}]; \mathbf{W}_2), \quad (12)$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_3[\hat{\mathbf{c}}_t; \mathbf{h}_t]), \quad (13)$$

where  $t = 1, 2, \dots, L_y$ ,  $\mathbf{e}_{t-1} \in \mathbb{R}^{d_e}$  is the embedding of the  $(t - 1)$ -th predicted word wherein  $\mathbf{e}_0$  is the embedding of the start token,  $\hat{\mathbf{c}}_t \in \mathbb{R}^d$  is the aggregated vector for  $\mathbf{h}_t \in \mathbb{R}^d$  from the memory bank  $[\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \dots, \tilde{\mathbf{m}}_{L_x}]$ , and  $\tilde{\mathbf{h}}_t \in \mathbb{R}^d$  is the attentional vector at time step  $t$ .

Consequently, the predicted probability distribution over the predefined vocabulary  $\mathcal{V}$  for current step is computed by:

$$P_v(y_t | \mathbf{y}_{t-1}, \mathbf{x}, \mathbf{t}) = \text{softmax}(\mathbf{W}_v \tilde{\mathbf{h}}_t + \mathbf{b}_v), \quad (14)$$

where  $\mathbf{y}_{t-1} = [y_1, \dots, y_{t-1}]$  is the previous predicted word sequence, and  $\mathbf{b}_v \in \mathbb{R}^{|\mathcal{V}|}$  is a learnable parameter vector.

Before generating the predicted word, a copy mechanism is adopted to efficiently exploit the in-text information and to strengthen the extraction capability of our model. We follow See, Liu, and Manning (2017) and first calculate a soft switch between generating from the vocabulary and copying from the source context  $\mathbf{x}$  at time step  $t$ :

$$g_t = \sigma(\mathbf{w}_g^T \tilde{\mathbf{h}}_t + b_g), \quad (15)$$

where  $\mathbf{w}_g \in \mathbb{R}^d$  is a learnable parameter vector and  $b_g$  is a learnable parameter scalar. Eventually, we get the final predicted probability distribution over the dynamic vocabulary  $\mathcal{V} \cup \mathcal{X}$ , where  $\mathcal{X}$  are all words appearing in the source context. For simplicity, we use  $P_v(y_t)$  and  $P_{final}(y_t)$  to denote  $P_v(y_t | \mathbf{y}_{t-1}, \mathbf{x}, \mathbf{t})$  and  $P_{final}(y_t | \mathbf{y}_{t-1}, \mathbf{x}, \mathbf{t})$  respectively:

$$P_{final}(y_t) = (1 - g_t)P_v(y_t) + g_t \sum_{i: x_i = y_t} \hat{\alpha}_{t,i}, \quad (16)$$

where  $\hat{\alpha}_{t,i}$  is the normalized attention score between  $\mathbf{h}_t$  and  $\tilde{\mathbf{m}}_i$ . For all out-of-vocabulary (OOV) words (i.e.,  $y_t \notin \mathcal{V}$ ), we set  $P_v(y_t)$  as zero. Similarly, if word  $y_t$  does not appear in the source context  $\mathbf{x}$  (i.e.,  $y_t \notin \mathcal{X}$ ), the copy probability  $\sum_{i: x_i = y_t} \hat{\alpha}_{t,i}$  is set as zero.

## Training

We use the negative log likelihood loss to train our model:

$$\mathcal{L} = - \sum_{t=1}^{L_y} \log P_{final}(y_t | \mathbf{y}_{t-1}, \mathbf{x}, \mathbf{t}; \Theta), \quad (17)$$

where  $L_y$  is the length of target keyphrase  $\mathbf{y}$  and  $y_t$  is the  $t$ -th target word in  $\mathbf{y}$ , and  $\Theta$  represents all the learnable parameters.

## Experiment Settings

The keyphrase prediction performance is first evaluated by comparing our model with the popular extractive methods and the state-of-the-art generative methods on five real-world benchmarks. Then, comparative experiments of different title length ratios are performed on our model and CopyRNN for further model exploration. Finally, an ablation study and a case study are conducted to better understand and interpret our model.

The experiment results lead to the following findings:

- Our model outperforms the state-of-the-art models on all the five benchmark datasets for both present and absent keyphrase prediction.
- Our model consistently improves the performance on various title length ratios and obtains relative higher improvement gains for both very low and very high title length ratios.
- The title-guided encoding part and the copy part are consistently effective in both present and absent keyphrase prediction tasks.

We implement the models using PyTorch (Paszke et al. 2017) on the basis of the OpenNMT-py system (Klein et al. 2017).

## Training Dataset

Because of the public accessibility, many commonly-used scientific publication datasets are used to evaluate the explored KG methods. This study also focuses on generating keyphrases from scientific publications. For all the generative models (i.e. our TG-Net model as well as all the encoder-decoder baselines), we choose the largest publicly available keyphrase generation dataset **KP20k** constructed by Meng et al. (2017) as the training dataset. **KP20k** consists of a large amount of high-quality scientific publications from various computer science domains. Totally 567,830 articles are collected in this dataset, where 527,830 for training, 20,000 for validation, and 20,000 for testing. Both the validation set and testing set are randomly selected. Since the other commonly-used datasets are too small to train a reliable generative model, we only train these generative models on **KP20k** and then test the trained models on all the testing part of the datasets listed in Table 2. As for the traditional supervised extractive baseline, we follow Meng et al. (2017) and use the dataset configuration shown in Table 2. To avoid the out-of-memory problem, for **KP20k**, we use the validation set to train the traditional supervised extractive baseline.

## Testing Datasets

Besides **KP20k**, we also adopt other four widely-used scientific datasets for comprehensive testing, including **Inspec** (Hulth 2003), **Krapivin** (Krapivin, Autaeu, and Marchese 2009), **NUS** (Nguyen and Kan 2007), and **SemEval-2010** (Kim et al. 2010). Table 2 summarizes the statistics of each testing dataset.

| Dataset             | Total   | Training | Testing |
|---------------------|---------|----------|---------|
| <b>Inspec</b>       | 2,000   | 1500     | 500     |
| <b>Krapivin</b>     | 2,304   | 1904     | 400     |
| <b>NUS</b>          | 211     | FFCV     | 211     |
| <b>SemEval-2010</b> | 288     | 188      | 100     |
| <b>KP20k</b>        | 567,830 | 20,000   | 20,000  |

Table 2: The statistics of testing datasets. The “Training” means the training part for the traditional supervised extractive baseline. The “FFCV” represents five-fold cross validation. The “Testing” means the testing part for all models.

## Implementation Details

For all datasets, the main body is the abstract, and the context is the concatenation of the title and the abstract. During preprocessing, various operations are performed including lowercasing, tokenizing by CoreNLP (Manning et al. 2014), and replacing all the digits with the symbol  $\langle digit \rangle$ . We define the vocabulary  $\mathcal{V}$  as the 50,000 most frequent words.

We set the embedding dimension  $d_e$  to 100, the hidden size  $d$  to 256, and  $\lambda$  to 0.5. All the initial states of GRU cells are set as zero vectors except that  $\mathbf{h}_0$  is initialized as  $[\vec{m}_{L_x}; \vec{m}_1]$ . We share the embedding matrix among the context words, the title words, and the target keyphrase words. All the trainable variables including the embedding matrix

are initialized randomly with uniform distribution in  $[-0.1, 0.1]$ . The model is optimized by Adam (Kingma and Ba 2015) with batch size = 64, initial learning rate = 0.001, gradient clipping = 1, and dropout rate = 0.1. We decay the learning rate into the half when the evaluation perplexity stops dropping. Early stopping is applied when the validation perplexity stops dropping for three continuous evaluations. During testing, we set the maximum depth of beam search as 6 and the beam size as 200. We repeat the experiments of our model three times using different random seeds and report the averaged results.

We do not remove any predicted single-word phrase in the post-processing for **KP20k** during testing, which is different from Meng et al. (2017), since our model is trained on this dataset and it can effectively learn the distribution of the single-word keyphrases. But for other testing datasets, we only keep the first predicted single-word phrase following Meng et al. (2017).

## Baseline Models and Evaluation Metric

For present keyphrase predicting experiment, we use two unsupervised models including TF-IDF and TextRank (Mihalcea and Tarau 2004), and one supervised model Maui (Medelyan, Frank, and Witten 2009) as our traditional extraction baselines. Besides, we also select CopyRNN (Meng et al. 2017) and CopyCNN (Zhang, Fang, and Weidong 2017), the two state-of-the-art encoder-decoder models with copy mechanism (Gu et al. 2016), as the baselines for present keyphrase prediction task. As for absent keyphrase prediction, since traditional extraction baselines cannot generate such keyphrases, we only choose CopyRNN and CopyCNN as the baseline models. For all baselines, we use the same setups as Meng et al. (2017) and Zhang, Fang, and Weidong (2017).

The *recall* and *F-measure* ( $F_1$ ) are employed as our metrics for evaluating these algorithms. *Recall* is the number of correctly predicted keyphrases over the total number of target keyphrases.  $F_1$  score is computed based on the *Recall* and the *Precision*, wherein *Precision* is defined as the number of correctly predicted keyphrases over the total predicted keyphrase number. Following Meng et al. (2017) and Zhang, Fang, and Weidong (2017), we also employ Porter Stemmer for preprocessing when determining whether two keyphrases are matched.

## Results and Analysis

### Present Keyphrase Predicting

In this section, we compare present keyphrase prediction ability of these models on the five real-world benchmark datasets. The F-measures at top 5 and top 10 predictions of each model are shown in Table 3.

From this table, we find that all the generative models significantly outperforms all the traditional extraction baselines. Besides, we also note that our TG-Net model achieves the best performance on all the datasets with significant margins. For example, on **KP20k** dataset, our model improves 9.4% ( $F_1@10$  score) than the best generative model CopyCNN. Compared to CopyRNN which also applies an RNN-

| Model    | Inspec            |                    | Krapivin          |                    | NUS               |                    | SemEval           |                    | KP20k             |                    |
|----------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|
|          | F <sub>1</sub> @5 | F <sub>1</sub> @10 | F <sub>1</sub> @5 | F <sub>1</sub> @10 | F <sub>1</sub> @5 | F <sub>1</sub> @10 | F <sub>1</sub> @5 | F <sub>1</sub> @10 | F <sub>1</sub> @5 | F <sub>1</sub> @10 |
| TF-IDF   | 0.221             | 0.313              | 0.129             | 0.160              | 0.136             | 0.184              | 0.128             | 0.194              | 0.102             | 0.162              |
| TextRank | 0.223             | 0.281              | 0.189             | 0.162              | 0.195             | 0.196              | 0.176             | 0.187              | 0.175             | 0.147              |
| Maui     | 0.040             | 0.042              | 0.249             | 0.216              | 0.249             | 0.268              | 0.044             | 0.039              | 0.270             | 0.230              |
| CopyRNN  | 0.278             | 0.342              | 0.311             | 0.266              | 0.334             | 0.326              | 0.293             | 0.304              | 0.333             | 0.262              |
| CopyCNN  | 0.285             | 0.346              | 0.314             | 0.272              | 0.342             | 0.330              | 0.295             | 0.308              | 0.351             | 0.288              |
| TG-Net   | <b>0.315</b>      | <b>0.381</b>       | <b>0.349</b>      | <b>0.295</b>       | <b>0.406</b>      | <b>0.370</b>       | <b>0.318</b>      | <b>0.322</b>       | <b>0.372</b>      | <b>0.315</b>       |
| % gain   | 10.5%             | 10.1%              | 11.1%             | 8.5%               | 18.7%             | 12.1%              | 7.8%              | 4.5%               | 6.0%              | 9.4%               |

Table 3: Present keyphrase predicting results on all test datasets. “% gain” is the improvement gain over CopyCNN.

| Model   | Inspec       |              | Krapivin     |              | NUS          |              | SemEval      |              | KP20k        |              |
|---------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
|         | R@10         | R@50         | R@10         | R@50         | R@10         | R@50         | R@10         | R@50         | R@10         | R@50         |
| CopyRNN | 0.047        | 0.100        | 0.113        | 0.202        | 0.058        | 0.116        | 0.043        | 0.067        | 0.125        | 0.211        |
| CopyCNN | 0.050        | 0.107        | 0.119        | 0.205        | 0.062        | 0.120        | 0.044        | 0.074        | 0.147        | 0.225        |
| TG-Net  | <b>0.063</b> | <b>0.115</b> | <b>0.146</b> | <b>0.253</b> | <b>0.075</b> | <b>0.137</b> | <b>0.045</b> | <b>0.076</b> | <b>0.156</b> | <b>0.268</b> |
| % gain  | 26.0%        | 7.5%         | 22.7%        | 23.4%        | 21.0%        | 14.2%        | 2.3%         | 2.7%         | 6.1%         | 19.1%        |

Table 4: Absent keyphrase predicting results on all test datasets. “% gain” is the improvement gain over CopyCNN.

based framework, our model improves about 20.2%. The results show that our model obtains much stronger keyphrase extraction ability than CopyRNN and CopyCNN.

### Absent Keyphrase Predicting

In this setting, we consider the absent keyphrase predicting ability which requires the understanding of the semantic meaning of the context. Only the absent target keyphrases and the absent predictions are preserved for this evaluation. Generally, recalls at top 10 and top 50 predictions are engaged as the metrics to evaluate how many absent target keyphrases are correctly predicted.

The performance of all models is listed in Table 4. It is observed that our TG-Net model consistently outperforms the previous sequence-to-sequence models on all the datasets. For instance, our model exceeds 19.1% (R@50 score) on **KP20k** than the state-of-the-art model CopyCNN. Overall, the results indicate that our model is able to capture the underlying semantic meaning of the context content much better than these baselines, as we have anticipated.

### Keyphrase Predicting on Various Title Length Ratios

To find out how our title incorporation influences the prediction ability, we compare the keyphrase predicting ability of two RNN-based models (i.e., our model and CopyRNN) on different title length ratios. The title length ratio is defined as the title length over the context length. This analysis is based on the **KP20k** testing dataset. In view of the title length ratio, we preprocess the testing set into five groups (i.e., <3%, 3%-6%, 6%-9%, 9%-12% and >12%). Then, the present keyphrase prediction results (F1@5 measure) and the improvement gain on each group are depicted in Figure 3.

In Figure 3(a), we notice that both CopyRNN and our TG-Net model generally perform better when the title length ra-

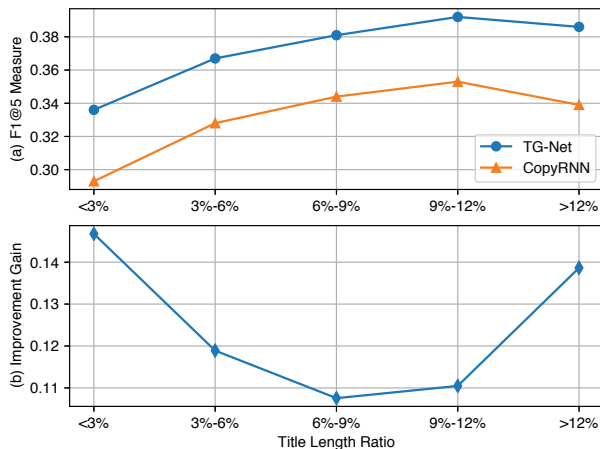


Figure 3: Present keyphrase predicting ability (F1@5 measure) on various title length ratios.

tio is higher. One possible explanation is that when the title is long, it conveys substantial salient information of the abstract. Therefore, the chance for the models to attend to the core information is enhanced, which leads to the observed situation. This figure also shows that both TG-Net and CopyRNN get worse performance on >12% group than 9%-12% group. The main reason is that there exist some data with a short abstract in >12% group, which leads to the lack of enough context information for correctly generating all keyphrases.

In Figure 3(b), we find that our TG-Net consistently improves the performance with a large margin on five testing groups, which again indicates the effectiveness of our model. In a finer perspective, we note that the improvement gain is higher on the lowest (i.e., <3%) and the highest (i.e.,



|   |
|---|
| <p><b>Title:</b> Exponential stability of switched stochastic delay systems with <b>non-linear uncertainties</b></p> <p><b>Abstract:</b> This article considers the robust <b>exponential stability</b> of uncertain switched stochastic systems with time-delay. Both almost sure (sample) stability and stability in mean square are investigated. Based on Lyapunov functional methods and <b>linear matrix inequality</b> techniques, new criteria for exponential robust stability of switched stochastic delay systems with <b>non-linear uncertainties</b> are derived in terms of linear matrix inequalities and <b>average dwell-time</b> conditions. Numerical examples are also given to illustrate the results.</p> |
| <p>(a) Present Keyphrases</p> <p><b>Target:</b> {stochastic systems; non-linear uncertainties; exponential stability; linear matrix inequality; average dwell-time}</p> <p><b>CopyRNN:</b> 1. <b>linear matrix inequality</b>, 2. switched stochastic systems, 3. robust stability, 4. <b>exponential stability</b>, 5. <b>average dwell-time</b></p> <p><b>TG-Net:</b> 1. <b>exponential stability</b>, 2. switched stochastic systems, 3. <b>average dwell-time</b>, 4. <b>non-linear uncertainties</b>, 5. <b>linear matrix inequality</b></p>   |
| <p>(b) Absent Keyphrases</p> <p><b>Target:</b> {switched systems; time-delay system}</p> <p><b>CopyRNN:</b> 1. <b>switched systems</b>, 2. switched delay systems, 3. robust control, 4. uncertain systems, 5. switched stochastic stochastic systems</p> <p><b>TG-Net:</b> 1. almost sure stability, 2. <b>switched systems</b>, 3. <b>time-delay systems</b>, 4. mean square stability, 5. uncertain systems</p>  |

Figure 4: A prediction example of CopyRNN and TG-Net. The top 5 predictions are compared and the correct predictions are highlighted in bold.

>12%) title length ratio groups. In >12% group, the title plays a more important role than in other groups, and consequently our model benefits more by not only explicitly emphasizing the title information itself, but also utilizing it to guide the encoding of information in the main body. As for <3% group, the effect of such a short title is small on the latter part of the context in CopyRNN because of the long distance. However, our model explicitly employs the title to guide the encoding of each context word regardless of the distance, which utilizes the title information much more sufficiently. Consequently, our model achieves much higher improvement in this group. While we only display the results of present keyphrase prediction, the absent keyphrase predicting task gets the similar results.

### Ablation Study

We also perform an ablation study on **Krapivin** for better understanding the contributions of the main parts of our model. For a comprehensive comparison, we conduct this study on both present keyphrase prediction and absent keyphrase prediction.

As shown in Table 5, after we remove the title-guided part and only reserve the sequence encoding for the context (i.e., -title), both the present and absent keyphrase prediction performance become obviously worse, indicating that our title-guided context encoding is consistently critical for both present and absent keyphrase generation tasks. We also investigate the effect of removing the copy mechanism (i.e., -copy) from our TG-Net. From Table 5, we notice that the scores decrease dramatically on both present and absent keyphrase prediction, which demonstrates the effectiveness of the copy mechanism in finding important parts of the context.

### Case Study

A keyphrase prediction example for a paper about the exponential stability of uncertain switched stochastic delay systems is shown in Figure 4. To be fair, we also only compare the RNN-based models (i.e., TG-Net and CopyRNN). For present keyphrase, we find that a present keyphrase “non-

| Model  | Present      |              | Absent       |              |
|--------|--------------|--------------|--------------|--------------|
|        | F1@5         | F1@10        | R@10         | R@50         |
| TG-Net | <b>0.349</b> | <b>0.295</b> | <b>0.146</b> | <b>0.253</b> |
| -title | 0.334        | 0.288        | 0.142        | 0.240        |
| -copy  | 0.306        | 0.281        | 0.127        | 0.216        |

Table 5: Ablation study on **Krapivin** dataset.

linear uncertainties”, which is a title phrase, is correctly predicted by our TG-Net, while CopyRNN fails to do so. As for absent keyphrase, we note that CopyRNN fails to predict the absent keyphrase “time-delay systems”. But our TG-Net can effectively utilize the title information “stochastic delay systems” to locate the important abstract information “stochastic systems with time-delay” and then successfully generate this absent keyphrase. These results exhibit that our model is capable of capturing the title-related core information more effectively and achieving better results in predicting present and absent keyphrases.

### Conclusion

In this paper, we propose a novel TG-Net for keyphrase generation task, which explicitly considers the leading role of the title to the overall document main body. Instead of simply concatenating the title and the main body as the only source input, our model explicitly treats the title as an extra query-like input to guide the encoding of the context. The proposed TG-Net is able to sufficiently leverage the highly summative information in the title to guide keyphrase generation. The empirical experiment results on five popular real-world datasets exhibit the effectiveness of our model for both present and absent keyphrase generation, especially for a document with very low or very high title length ratio. One interesting future direction is to explore more appropriate evaluation metrics for the predicted keyphrases instead of only considering the exact match with the human labeled keyphrases as the current *recall* and *F-measure* do.

## Acknowledgments

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 and No. CUHK 14210717 of the General Research Fund), and Microsoft Research Asia (2018 Microsoft Research Asia Collaborative Research Award). We would like to thank Jingjing Li, Hou Pong Chan, Piji Li and Lidong Bing for their comments.

## References

- [2015] Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.
- [2011] Berend, G. 2011. Opinion expression mining by exploiting keyphrase extraction. In *IJCNLP*, 1162–1170.
- [2014] Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, 1724–1734.
- [2017] Dauphin, Y. N.; Fan, A.; Auli, M.; and Grangier, D. 2017. Language modeling with gated convolutional networks. In *ICML*, 933–941.
- [2017] Florescu, C., and Caragea, C. 2017. A position-biased pagerank algorithm for keyphrase extraction. In *AAAI Student Abstracts*, 4923–4924.
- [2018] Florescu, C., and Jin, W. 2018. Learning feature representations for keyphrase extraction. In *AAAI Student Abstracts*.
- [2018] Gao, Y.; Bing, L.; Li, P.; King, I.; and Lyu, M. R. 2018. Generating distractors for reading comprehension questions from real examinations. *arXiv preprint arXiv:1809.02768*.
- [2017] Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. In *ICML*, 1243–1252.
- [2017] Gollapalli, S. D.; Li, X.; and Yang, P. 2017. Incorporating expert knowledge into keyphrase extraction. In *AAAI*, 3180–3187.
- [2016] Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*, volume 1, 1631–1640.
- [2018] Hai, Y., and Lu, W. 2018. Semi-supervised learning for neural keyphrase generation. *arXiv preprint arXiv:1808.06773*.
- [2006] Hulth, A., and Megyesi, B. B. 2006. A study on automatically extracted keywords in text categorization. In *COLING and ACL*, 537–544.
- [2003] Hulth, A. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *EMNLP*, 216–223.
- [1999] Jones, S., and Staveley, M. S. 1999. Phrasier: a system for interactive document retrieval using keyphrases. In *SIGIR*, 160–167.
- [2018] Jun, C.; Xiaoming, Z.; Yu, W.; Zhao, Y.; and Zhoujun, L. 2018. Keyphrase generation with correlation constraints. *arXiv preprint arXiv:1808.07185*.
- [2010] Kim, S. N.; Medelyan, O.; Kan, M.-Y.; and Baldwin, T. 2010. Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, 21–26.
- [2015] Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.
- [2017] Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; and Rush, A. 2017. Opennmt: Open-source toolkit for neural machine translation. In *ACL System Demonstrations*, 67–72.
- [2009] Krapivin, M.; Autaeu, A.; and Marchese, M. 2009. Large dataset for keyphrases extraction. Technical report, University of Trento.
- [2010] Li, D.; Li, S.; Li, W.; Wang, W.; and Qu, W. 2010. A semi-supervised key phrase extraction approach: Learning from title phrases through a document semantic network. In *ACL Short*, 296–300.
- [2011] Liu, Z.; Chen, X.; Zheng, Y.; and Sun, M. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *CoNLL*, 135–144.
- [2017] Luan, Y.; Ostendorf, M.; and Hajishirzi, H. 2017. Scientific information extraction with semi-supervised neural tagging. In *EMNLP*, 2641–2651.
- [2015] Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, 1412–1421.
- [2014] Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; and McClosky, D. 2014. The stanford corenlp natural language processing toolkit. In *ACL System Demonstrations*, 55–60.
- [2009] Medelyan, O.; Frank, E.; and Witten, I. H. 2009. Human-competitive tagging using automatic keyphrase extraction. In *EMNLP*, 1318–1327.
- [2017] Meng, R.; Zhao, S.; Han, S.; He, D.; Brusilovsky, P.; and Chi, Y. 2017. Deep keyphrase generation. In *ACL*, volume 1, 582–592.
- [2004] Mihalcea, R., and Tarau, P. 2004. Texttrank: Bringing order into text. In *EMNLP*.
- [2017] Nema, P.; Khapra, M. M.; Laha, A.; and Ravindran, B. 2017. Diversity driven attention model for query-based abstractive summarization. In *ACL*, volume 1, 1063–1072.
- [2007] Nguyen, T. D., and Kan, M.-Y. 2007. Keyphrase extraction in scientific publications. In *ICADL*, 317–326.
- [2010] Nguyen, T. D., and Luong, M.-T. 2010. Wingnus: Keyphrase extraction utilizing document logical structure. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, 166–169.
- [2017] Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. In *NIPS-W*.
- [2017] See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*, volume 1, 1073–1083.
- [2017] Song, L.; Wang, Z.; and Hamza, W. 2017. A unified query-based generative model for question generation and question answering. *arXiv preprint arXiv:1709.01058*.
- [2014] Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*, 3104–3112.
- [2008] Wan, X., and Xiao, J. 2008. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, 855–860.
- [2017] Wang, W.; Yang, N.; Wei, F.; Chang, B.; and Zhou, M. 2017. Gated self-matching networks for reading comprehension and question answering. In *ACL*, volume 1, 189–198.
- [1999] Witten, I. H.; Paynter, G. W.; Frank, E.; Gutwin, C.; and Nevill-Manning, C. G. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, 254–255.



- [2016] Zhang, Q.; Wang, Y.; Gong, Y.; and Huang, X. 2016. Keyphrase extraction using deep recurrent neural networks on twitter. In *EMNLP*, 836–845.
- [2017] Zhang, Y.; Fang, Y.; and Weidong, X. 2017. Deep keyphrase generation with a convolutional sequence to sequence model. In *ICSAI*, 1477–1485.