Incorporating Linguistic Constraints into Keyphrase Generation

Jing Zhao and **Yuxiang Zhang** [*]

Civil Aviation University of China, Tianjin, China
jing.zhao.me@outlook.com, yxzhcn@sina.com

Abstract

Keyphrases, that concisely describe the high-level topics discussed in a document, are very useful for a wide range of natural language processing tasks. Though existing keyphrase generation methods have achieved remarkable performance on this task, they generate many overlapping phrases (including sub-phrases or super-phrases) of keyphrases. In this paper, we propose the parallel Seq2Seq network with the coverage attention to alleviate the overlapping phrase problem. Specifically, we integrate the linguistic constraints of keyphrases into the basic Seq2Seq network on the source side, and employ the multi-task learning framework on the target side. In addition, in order to prevent from generating overlapping phrases with correct syntax, we introduce the coverage vector to keep track of the attention history and to decide whether the parts of source text have been covered by existing generated keyphrases. The experimental results show that our method can outperform the state-of-the-art CopyRNN on scientific datasets, and is also more effective in news domain.

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 tasks such as information extraction (Wan and Xiao, 2008), text summarization (Zhang et al., 2017) and question answering (Tang et al., 2017). However, the performance of existing methods is still far from satisfactory (Hasan and Ng, 2014). The main reason is that it is very challenging to determine whether a phrase or sets of phrases can accurately capture main topics that are presented in the document.

Existing approaches for keyphrase prediction can be broadly divided into extraction and generation methods. The conventional extraction methods directly select important consecutive words or phrases from the target document as keyphrases. This means that the extracted keyphrases must appear in the target document. In comparison with extraction methods, the generation methods choose keyphrases from a predefined vocabulary regardless of whether the generated keyphrases appear in the target document. CopyRNN (Meng et al., 2017) is the first to employ the sequence-to-sequence (Seq2Seq) framework (Sutskever et al., 2014) to generate keyphrases for documents. This method is able to predict absent keyphrases that do not appear in the target document.

Following the CopyRNN, a few extensions of Seq2Seq framework have been proposed to help better generate keyphrases. Through analyzing the results generated by these approaches, we find out that there are many overlapping phrases of correct (author-labeled) keyphrases. For example, in experimental results of CopyRNN, the author-labeled keyphrases are “Internet” and “Distributed decision” but the predicted are “Internet held” and “Distributed”, respectively. There are two shortcomings that lie in the overlapping phrases. First, the correct keyphrase is not generated but its overlapping phrases are predicted as keyphrases. Second, the existing generation approaches often predict the keyphrase and its overlapping phrases as keyphrases. However, the overlapping phrases of keyphrases are not keyphrases in most cases. The more accurate description for this overlapping problem and shortcomings will be given in the next section, including the problem formulation and seriousness found in experimental results of the state-of-the-art CopyRNN.

[*] Corresponding author

Sub-problems and formulations			No.	Seriousness of the problem (top-k, $k=10$)				
				$\lvert P_i\rvert/\lvert O_l\rvert$ (%)	$\lvert P_i^n\rvert/\lvert O_l^n\rvert$ (%)			
					$n=1$	$n=2$	$n=3$	$n\geq 4$
$p \notin O_k$	$p_b \in O_k$	-	1	6.62	0	2.69	21.49	47.15
	$p_u \in O_k$	-	2	11.10	23.98	3.30	1.71	0.81
$p \in O_k$	$p_b \in O_k$	$Top(p) > Top(p_b)$	3	5.58	0	5.09	17.44	17.36
		$Top(p) < Top(p_b)$	4	7.25	0	4.73	28.89	17.46
	$p_u \in O_k$	$Top(p) > Top(p_u)$	5	1.41	0.85	2.39	0.63	0.24
		$Top(p) < Top(p_u)$	6	10.77	9.78	14.84	5.53	1.41
Total				**42.73**	**34.61**	**33.04**	**75.69**	**84.43**

Table 1: Problem formulation and seriousness in experimental results of CopyRNN.

In this paper, we propose a parallel Seq2Seq network (**ParaNet**) with the coverage attention to alleviate the overlapping phrase problem. Specifically, we exploit two standalone encoders to encode separately the source text and syntactic constraints into network on the source side, and then applies multi-task learning framework to generate the keyphrases and part-of-speech (POS) tags for words in keyphrases on the target side. Most of keyphrases are noun phrases and they commonly consist of nouns and adjectives. The syntactic constraints are helpful to prevent from generating the overlapping phrases of keyphrases that are not noun phrases, e.g., "internet held" (which contains a verb). In addition, in order to prevent from generating overlapping phrases of keyphrases with correct syntax, we introduce the coverage vector (proposed in (Tu et al., 2016)) to keep track of the attention history and to decide whether the parts of source text have been covered by the existing generated keyphrases.

The remaining of this paper is organized as follows. In the next section, we analyze the overlapping phrase problem in existing generation methods. We summarize related methods to keyphrase prediction, especially for keyphrase generation in Section 3. The proposed method is presented in Section 4. Finally, we show the experiments and results before concluding the paper.

2 Analysis of the Overlapping Problem

In this section, we first formalize the overlapping phrase problem, and then present its seriousness by analyzing statistics obtained from CopyRNN.

Let $p = w_i w_{i+1} ... w_{i+m}$ be a phrase with lengths $m+1$ over a finite word dictionary D, i.e., $w_i \in D$. we define the phrase $p_b = w_{i+j} w_{i+j+1} ... w_{i+j+k}$ $(j \geq 0, j+k \leq m)$ as a sub-phrase of p. Conversely, we define the phrase p as a super-phrase of p_b and denote the super-phrase of p as p_u. Overlapping relations exist between phrase p and its sub/super-phrase p_b/p_u. Let O_l be a set of author-labeled keyphrases, and O_k be a set of the generated keyphrases at top-k predictions, in which each generated phrase may be correct or incorrect. We assume that p is an author-labeled keyphrase, i.e., $p \in O_l$, and its sub-phrase p_b and super-phrase p_u are not keyphrases, i.e., $p_b, p_u \notin O_l$. Let $Top(p_x)$ be the rank of predicted keyphrase p_x in O_k. $Top(p) > Top(p_x)$ means that the rank of $Top(p)$ is higher than $Top(p_x)$.

The overlapping phrase problem can be divided into two main problems according to whether p is generated at the top-k results. These two problems are further subdivided into six sub-problems, formulated as shown in Table 1. The formulations No.1-2 shown in Table 1 mean that the author-labeled keyphrase p is not predicted, and only one of its sub-phrases p_b or super-phrases p_u is generated. The formulations No.3-6 in Table 1 mean that the author-labeled keyphrase p and one of its sub-phrases p_b or super-phrases p_u are generated. In addition, $Top(p) < Top(p_b/p_u)$ is worse than $Top(p) > Top(p_b/p_u)$. Note that p, p_b and p_u are rarely generated simultaneously.

We next present the seriousness of this problem through analyzing statistics obtained from experimental results of CopyRNN on dataset KP20k. We first calculate the proportion of the keyphrases suffering from the i-th sub-problem in all correct keyphrases, i.e., $|P_i|/|O_l|$, where P_i is defined as $P_i = \{p | p \in O_l \wedge p$ suffers from the i-th sub-problem$\}$, $|P_i|$ and $|O_l|$ are respectively the size of P_i and O_l. We select top-k $(k=10)$ phrases generated by CopyRNN as the final predictions. As the results of $|P_i|/|O_l|$ shown in Table 1, a total of 42.73% keyphrases suffer from this problem.

In addition, we calculate the proportion of the keyphrases with the length n which suffer from the i-th sub-problem in all correct keyphrases with the same length, i.e., $|P_i^n|/|O_l^n|$, where P_i^n and O_l^n are the subsets of P_i and O_l, respectively, in which the length of each keyphrase is n (i.e., keyphrase is n-gram). Table 1 also shows the seriousness of the sub-problems of overlapping phrase problem with varying n of n-grams. As the results show, we can observe that the long keyphrases can easily suffer from the sub-phrase problem (i.e., $p_b \in O_k$) and the short keyphrases can easily suffer from the super-phrase problem (i.e., $p_u \in O_k$ in Table 1).

Although the overlapping problem restricts the performance of existing methods, it also gives us an opportunity to help better generate keyphrases as the overlapping phrases are often very close to the correct keyphrases.

3 Related Works

As mentioned in Section 1, existing approaches for keyphrase prediction can be broadly divided into extraction and generation methods. The extraction methods can be further classified into supervised and unsupervised approaches. The supervised approaches treat keyphrase extraction as a binary classification task, in which a learning model is trained on 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 directly treat keyphrase extraction as a ranking problem, scoring each candidate using different kinds of techniques such as clustering (Liu et al., 2009), or graph-based ranking (Mihalcea and Tarau, 2004; Wan and Xiao, 2008).

This work is mainly related to keyphrase generation approaches which have been proven to be effective in the keyphrase prediction task. Following CopyRNN (Meng et al., 2017) which is the first to generate absent keyphrases using Seq2Seq framework, the few extensions have been proposed to help better generate keyphrases.

In CopyRNN, model training heavily relies on massive amounts of labeled data, which is often unavailable especially for the new domains. To solve this problem, Ye and Wang (2018) proposed a semi-supervised keyphrase generation model by leveraging both abundant unlabeled data and limited labeled data. CopyRNN does not model the one-to-many relationship between the document and keyphrases. Therefore, keyphrase generation only depends on the source document and ignores constraints on the correlation among keyphrases. To overcome this drawback, Chen et al. (2018) proposed a Seq2Seq network with correlation constraints for keyphrase generation. Chen et al. (2019) proposed a title-guided Seq2Seq network to use title of source text to improve performance. However, these methods did not consider the linguistic constraints of keyphrases.

4 Methodology

4.1 Problem Definition

Given a text dataset $D = \{\mathbf{x}_i, \mathbf{p}_i\}_{i=1}^{N}$, where $\mathbf{x}_i$ is a source document, $\mathbf{p}_i = \{p_{i,j}\}_{j=1}^{M_i}$ is the keyphrase set of $\mathbf{x}_i$, and N is the number of documents. Both the document $\mathbf{x}_i$ and keyphrase $p_{i,j}$ are sequences of words, denoted as $\mathbf{x}_i = (x_1^{(i)}, x_2^{(i)}, ..., x_{L_i}^{(i)})$ and $p_{i,j} = (y_1^{(i,j)}, y_2^{(i,j)}, ..., y_{L_{ij}}^{(i,j)})$, where L_i and L_{ij} are the length of word sequence of $\mathbf{x}_i$ and $p_{i,j}$. The goal of a keyphrase generation is to design a model to map each document $\mathbf{x}$ into the keyphrase set $\mathbf{p}$.

4.2 Model Overview

Figure 1 illustrates the overview of the proposed method. The method consists of two components, which are the parallel encoders and decoders. The parallel encoders consist of the word encoder and syntactic information encoder, which are used to compress the source text and its syntactic information into the hidden vectors. The parallel decoders contain the keyphrase decoder and POS tag decoder, which are different decoders and used to generate the keyphrases and POS tags of words in keyphrases. During the training process, these two tasks boost each other providing strong representation for source text. In addition, we employ the coverage attention to alleviate generating the overlapping phrases of keyphrases.

4.3 Basic Seq2Seq Model

Our approach is based on a Seq2Seq framework which consists of an encoder and a decoder. Both the encoder and decoder are implemented with recurrent neural networks (RNN). The encoder converts the variable-length source word sequence

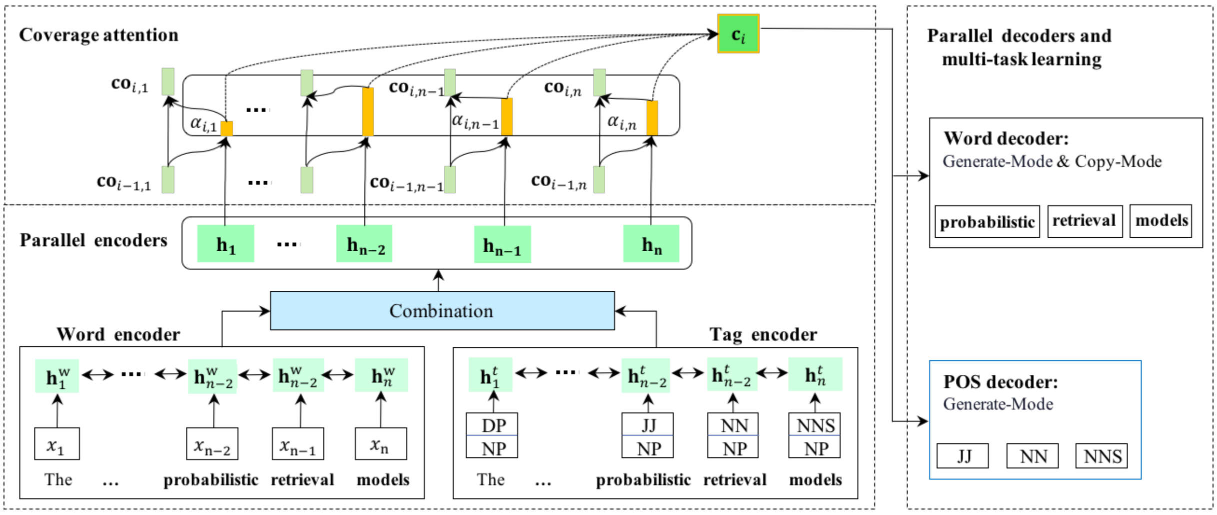
Coverage attention
Parallel encoders
Combination
Word encoder
Tag encoder
The ... probabilistic retrieval models
DP NP JJ NP NN NP NNS NP
Parallel decoders and multi-task learning
Word decoder: Generate-Mode & Copy-Mode
probabilistic retrieval models
POS decoder: Generate-Mode
JJ NN NNS

Figure 1: The overview of the proposed approach.

$\mathbf{x} = (x_1, x_2, ..., x_L)$ into a set of hidden representation vector $\{\mathbf{h}_i\}_{i=1}^{L}$, by iterating the following equation:

$$\mathbf{h}_i = f_e(x_i, \mathbf{h}_{i-1}) \tag{1}$$

where where f_e is a non-linear function in encoder.

The decoder decompresses the context vector and generate the variable-length target keyphrase $\mathbf{y} = (y_1, y_2, ..., y_{L'})$ word by word, through the conditional language model:

$$p(y_i|y_{1,...,i-1}, \mathbf{x}) = g(y_{i-1}, \mathbf{s}_i, \mathbf{c}_i) \tag{2}$$

where g is a softmax function, and $\mathbf{s}_i$ is a decoder hidden vector calculated as:

$$\mathbf{s}_i = f_d(y_{i-1}, \mathbf{s}_{i-1}, \mathbf{c}_i) \tag{3}$$

where f_d is a non-linear function in decoder. $\mathbf{c}_i$ is a context vector, calculated as a weight sum over source hidden vector $\mathbf{h}$:

$$\mathbf{c}_i = \textstyle\sum_{j=1}^{L} \alpha_{i,j}\mathbf{h}_j \tag{4}$$

$$\alpha_{i,j} = \frac{exp(a(\mathbf{s}_{i-1}, \mathbf{h}_j))}{\sum_{k=1}^{L} exp(a(\mathbf{s}_{i-1}, \mathbf{h}_k))} \tag{5}$$

where $a(\mathbf{s}_{i-1}, \mathbf{h}_j)$ is an alignment function that measures the similarity between $\mathbf{s}_{i-1}$ and $\mathbf{h}_j$.

Pure generation mode can not predict keyphrase which consists of out-of-vocabulary words. Thus, Meng et al. (2017) first introduced a copy mechanism (Gu et al., 2016) to predict out-of-vocabulary by directly copying words from source text. Consequently, the probability of generating a target word y_i (i.e., Equ. 2) is modified as:

$$p(y_i|y_{<i}, \mathbf{x}) = p_g(y_i|y_{<i}, \mathbf{x}) + p_c(y_i|y_{<i}, \mathbf{x}) \tag{6}$$

where $y_{<i}$ represents $y_{1,...,i-1}$ and p_c is the probability of copying, calculated as:

$$p_c(y_i|y_{<i}, \mathbf{x}) = \frac{1}{Z} \sum_{j:x_j=y_i} exp(\phi(x_j)), y_i \in \mathcal{X}$$
$$\phi(x_j) = \sigma(\mathbf{h}_j^{\top}\mathbf{W}_c)\mathbf{s}_i \tag{7}$$

where σ is a non-linear function, $\mathcal{X}$ is the set of unique words in source text $\mathbf{x}$, $\mathbf{W}_c$ is a learned parameter matrix and Z is the sum for normalization.

4.4 Parallel Seq2Seq Model

Most of keyphrases are noun phrases which commonly consist of nouns and adjectives (Gollapalli and Caragea, 2014). Hence, the syntactic information is useful for improving keyphrase generation performance. Although conventional generation model is capable of implicitly learning the syntactic information from source text, it can not capture a lot of deep syntactic structural details (Shi et al., 2016). To overcome this shortcoming, we propose a parallel Seq2Seq model which deeply integrates the following additional syntactic information into the basic Seq2Seq model:

- POS tag: Keyphrases are commonly noun phrases with a specified part-of-speech (POS) patterns (Hulth, 2003). In supervised approaches for keyphrase extraction, POS tags assigned to words have been chosen as one type of important syntactic features, used to train the classifier (Hasan and Ng, 2014; Gollapalli et al., 2017). We incorporate the POS tags into Seq2Seq network to capture the syntactic combinations of keyphrases.

Sentence:	The	framework	is	useful	for	deciding	the	parameter	estimation	in	**probabilistic**	**retrieval**	**models**
POS tags:	DT	NN	VBZ	JJ	IN	VBG	DT	NN	NN	IN	JJ	NN	NNS
Phrase tags:	NP	NP	VP	ADJP	PP	VP	NP	NP	NP	PP	NP	NP	NP

Table 2: An example of word sequence with both POS and phrase tags.

- Phrase tag: Phrase tags assigned to words are also one type of important syntactic features in supervised extraction approaches, since the words in keyphrase commonly share the same phrase tags (Gollapalli et al., 2017). Therefore, we integrate the phrase tags into Seq2Seq network to capture the inherent syntactic structure of keyphrases.

We use Stanford Parser[1] (Finkel et al., 2005) to obtain the 32 POS tags and 16 phrase tags of words. An example is shown in Table 2 with both POS and phrase tags, and the author-labeled keyphrase is highlighted in bold.

4.4.1 Parallel Encoders

The proposed model encodes word sequence and tag sequences (including POS and phrase tags) in parallel. We use the RNN encoder to produce the set of word hidden vector $\{\mathbf{h}^w\}$ from the source document $\mathbf{x}$, and produce the set of syntactic tag hidden vector $\{\mathbf{h}^t\}$ from the POS and phrase tags. We create the look-up based embedding matrices for word, POS tag and phrase tag, and concatenate the embeddings of POS tag and phrase tag into a long vector as input of the tag encoder.

We employ two methods to combine the word and syntactic tag hidden vectors into a unified hidden vector $\mathbf{h}$. The first method is inspired by the Tree-LSTM (Tai et al., 2015), which can selectively incorporate the information from each child node. The cell and hidden vectors are calculated by following transition equations:

$$\mathbf{i}_i = \sigma(\mathbf{W}^i_w \mathbf{h}^w_i + \mathbf{W}^i_t \mathbf{h}^t_i) \tag{8}$$

$$\mathbf{f}^w_i = \sigma(\mathbf{W}^{fw}_w \mathbf{h}^w_i + \mathbf{W}^{fw}_t \mathbf{h}^t_i) \tag{9}$$

$$\mathbf{f}^t_i = \sigma(\mathbf{W}^{ft}_w \mathbf{h}^w_i + \mathbf{W}^{ft}_t \mathbf{h}^t_i) \tag{10}$$

$$\mathbf{o}_i = \sigma(\mathbf{W}^o_w \mathbf{h}^w_i + \mathbf{W}^o_t \mathbf{h}^t_i) \tag{11}$$

$$\mathbf{u}_i = \tanh(\mathbf{W}^u_w h^w_i + \mathbf{W}^u_t h^t_i) \tag{12}$$

$$\mathbf{c}_i = \mathbf{i}_i \odot \mathbf{u}_i + \mathbf{f}^w_i \odot \mathbf{c}^w_i + \mathbf{f}^t_i \odot \mathbf{c}^t_i \tag{13}$$

$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i) \tag{14}$$

where $\mathbf{c}^w_i$ and $\mathbf{c}^t_i$ are the cell vectors of word and tag, $\mathbf{h}^w_i$ and $\mathbf{h}^t_i$ are the hidden vectors of word and tag, and σ is the sigmoid function. Each of $\mathbf{i}_i$, $\mathbf{f}^w_i$, $\mathbf{f}^t_i$, $\mathbf{o}_i$ and $\mathbf{u}_i$ denotes an input gate, a forget gate of word, a forget gate of syntactic tag, an output gate, and a vector for updating the memory cell, respectively. More details are given in (Tai et al., 2015).

The second method is the line transformation followed by the hyperbolic tangent function:

$$\mathbf{h}_i = \tanh(\mathbf{W}^l_w \mathbf{h}^w_i + \mathbf{W}^l_t \mathbf{h}^t_i). \tag{15}$$

4.4.2 Parallel Decoders

The proposed method consists of two parallel decoders: keyphrase decoder and POS tag decoder. The keyphrase decoder is used to generate a set of keyphrases for documents. Although the keyphrase decoder also can learn syntactic structures of keyphrases to some extent, it fails to capture deep syntactic details. In order to supervise the syntactic combinations of keyphrase, the POS tag decoder is employed to generate a series of POS tags of words in keyphrases. Note that the POS tag decoder in our model serves as a training-assisted role and is not used in the testing.

The probability of predicting each POS tag of word is given as follows:

$$p(t_i|t_{<i}, \mathbf{x}) = g_t(t_{i-1}, \mathbf{s}^t_i, \mathbf{c}_i) \tag{16}$$

where g_t is a softmax function, $\mathbf{s}^t_i$ is a hidden vector of POS tag decoder.

4.5 Coverage Attention

Repetition is a common problem for the Seq2Seq models and is especially serious when generating text sequence, such as machine translation (Tu et al., 2016) and automatic text summarization (See et al., 2017). The reason for this is that the traditional attention mechanisms focus on calculating the attention weight of the current time step, ignoring the distribution of weights in history. There can be no doubt that existing Seq2Seq models for keyphrase generation also suffer from this problem, i.e., generating sub-phrases or super-phrases of keyphrases. We employ the coverage

[1]https://nlp.stanford.edu/software/lex-parser.shtml

Dataset	#PKPs	#AKPs	#Abs	#1-grams	#2-grams	#3-grams	#4-grams	#>4-grams
Inspec	3,564	1,349	500	510/100	1,743/548	910/399	275/180	126/122
Krapivin	1,299	1,040	400	256/101	700/631	254/233	74/55	15/20
NUS	1,333	1,128	211	434/167	632/576	204/234	53/88	10/63
SemEval	625	841	100	162/107	309/398	113/204	28/60	13/72
KP20k	66,468	39,055	20,000	26,249/6,076	26,755/19,883	10,486/9,196	2,312/2,708	666/1,192

Table 3: Summary of Datasets.

model, used in works (Tu et al., 2016; See et al., 2017), to alleviate this problem.

In the coverage model, we maintain a coverage vector $\mathbf{co}$ to help adjust the future attention through keeping track of the attention history, calculated as:

$$\mathbf{co}_{i,j} = \mathbf{co}_{i-1,j} + \alpha_{i,j} \tag{17}$$

where the coverage vector $\mathbf{co}_{i,j}$ is used to measure the attention coverage degree of word x_j at step i. More details are shown in (Tu et al., 2016; See et al., 2017).

Finally, we integrate coverage vector the attention mechanism, by modifying the alignment function in Equation (5) as:

$$\begin{aligned} & a(\mathbf{s}_{i-1}, \mathbf{h}_j, \mathbf{co}_{i-1,j}) = \\ & \mathbf{v}_c^\top \tanh(\mathbf{W}_s \mathbf{s}_{i-1} + \mathbf{W}_h \mathbf{h}_j + \mathbf{W}_{co} \mathbf{co}_{i-1,j}) \end{aligned} \tag{18}$$

where $\mathbf{v}_c$, $\mathbf{W}_s$, $\mathbf{W}_h$, and $\mathbf{W}_{co}$ are the learnable weight parameters.

4.6 Overall Loss Function

Given the set of data pairs $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, where $\mathbf{x}$ is the word sequence of the source text, $\mathbf{y}$ is the word sequence of its keyphrase, and y is the word of keyphrase $\mathbf{y}$. The loss function consists of two parts. The first is the negative log-likelihood of the target words in keyphrase, calculated as:

$$\mathcal{L}_w(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{L_i} log(p(y_k^i | y_{<k}^i, \mathbf{x}^i; \theta_w)) \tag{19}$$

where L_i is the length of keyphrase $\mathbf{y}$, and θ_w is the parameter of this task.

The second loss function is the negative log-likelihood of the POS tags of words in keyphrases, calculated as follows:

$$\mathcal{L}_t(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{L_i} log(p(t_k^i | t_{<k}^i, \mathbf{x}^i; \theta_t)) \tag{20}$$

where t is the POS tag, and θ_t are the parameter. The final goal is to jointly minimize the two losses with Adam optimizer (Kingma and Ba, 2015):

$$\mathcal{L} = (1-\lambda)\mathcal{L}_w + \lambda \mathcal{L}_t \tag{21}$$

where λ is a hyper-parameter to tune the impacts of the two tasks.

5 Experiment

5.1 Datasets

We use the dataset collected by Meng et al. (2017) from various online digital libraries, which contains about 568K articles[2]. Following Meng et al. (2017), we use about 530K articles for training the model, 20k articles for validating the model, and 20k articles (i.e., **KP20k**) for testing the model. Similar to Meng et al. (2017), we also test the model on four widely used public datasets from the computer science domain: Inspec (Hulth and Megyesi, 2006), Krapivin (Krapivin et al., 2009), NUS (Nguyen and Kan, 2007), and SemEval-2010 (Kim et al., 2010).

The datasets are summarized in Table 3 along with the number of present keyphrase (#PKPs), the number of absent keyphrase (#AKPs), the number of articles (#Abs.), the number of present/absent 1-grams, 2-grams, 3-grams, 4-grams and more than 4-grams (#>4-grams), in each collection.

5.2 Experimental Settings

In the training dataset, input text is the concatenation of the title and abstract of the scientific articles. Following the work (Meng et al., 2017), all numbers in text are mapped to a special token <digit>. The syntactic tags include 32 POS tags and 16 phrase tags. The size of word vocabulary is set to 50,000, the size of word embeddings is set to 150, and the size of embeddings of two syntactic tags is set to 50. All embeddings are randomly initialized with uniform distribution in [-0.1,0.1], and

[2]https://github.com/memray/seq2seq-keyphrase

Method	Inspec		Krapivin		NUS		SemEval		KP20k	
	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10
BL*	0.223	0.313	0.249	0.216	0.249	0.268	0.176	0.194	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
ConNet	0.265	0.321	0.309	0.256	0.336	0.329	0.294	0.302	0.325	0.257
ParaNet$_L$	0.289	0.353	0.326	0.277	0.354	0.342	0.307	0.303	0.351	0.282
ParaNet$_T$	0.292	0.355	0.327	0.281	**0.360**	0.349	**0.313**	0.309	0.357	0.287
ParaNet$_L$+CoAtt	0.292	0.354	**0.330**	0.279	0.357	0.342	0.308	0.306	0.355	0.283
ParaNet$_T$+CoAtt	**0.296**	**0.357**	0.329	**0.282**	**0.360**	**0.350**	0.311	**0.312**	**0.360**	**0.289**

Table 4: Comparisons of predicting present keyphrases on five scientific datasets.

learned during training. The size of hidden vector is fixed at 300. The weight parameter used to tune the impacts of the two tasks is set to $\lambda = 0.3$. The initial learning rate of Adam optimizer is set to 10^{-4}, and the dropout rate is set to 0.5. We use the beam search to generate multiple phrases. The max depth of beam search is set to 6, and the beam size is set to 200.

5.3 Comparative Methods

We compare our method with extraction and generation approaches. Extraction methods consist of three unsupervised and two supervised methods. Unsupervised extraction methods include TF-IDF, TextRank (Mihalcea and Tarau, 2004) and SingleRank (Wan and Xiao, 2008). Supervised extraction methods include Maui (Medelyan et al., 2009) and KEA (Witten et al., 1999). To clearly represent the experimental results, we select the best-performing method (BL*) from these extraction baselines with best-performing parameters for each dataset to compare with our method. The generation baselines are state-of-the-art CopyRNN (Meng et al., 2017) and ConNet, which inputs the concatenation of word embeddings and two syntactic tag embeddings into CopyRNN.

The proposed method includes four models: (1) ParaNet$_L$, using the hyperbolic tangent function (i.e., Equ. 15) to combine two hidden vectors of words and syntactic tag generated by encoder; (2) ParaNet$_T$, using the tree-LSTM to combine two hidden vectors; (3) ParaNet$_L$+CoAtt, ParaNet$_L$ with the coverage attention; (4) ParaNet$_T$+CoAtt, ParaNet$_T$ with the coverage attention.

5.4 Evaluation Metrics

Almost all previous works on keyphrase prediction use precision (P), recall (R), F1-score (F1) to evaluate the results (Manning et al., 2010).

$$P = \frac{\#_c}{\#_p}, \quad R = \frac{\#_c}{\#_l}, \quad F1 = \frac{2PR}{P+R}, \qquad (22)$$

where $\#_c$ is the number of correctly predicted keyphrases, $\#_p$ is the total number of predicted keyphrases, and $\#_l$ is the total number of author-labeled standard keyphrases. Following the study (Meng et al., 2017), we employ top-N macro-averaged F1-score (F1) for evaluating present keyphrases and recall (R) for evaluating absent keyphrases. We use Porter's stemmer[3] to remove words' suffix before determining the match of two keyphrases.

5.5 Results and Analysis

5.5.1 Prediction of Present Keyphrases

The experimental results are shown in Table 4, in which the F1 at top-5 and top-10 predictions are given and the best scores are highlighted in bold. We compare our method with the best-performing extractive method (BL*), which can only extract the keyphrases that appear in the source text (i.e., present keyphrases).

We first compare our proposed method with the conventional keyphrases extraction methods. The results show that even the worst one in our models (i.e., ParaNet$_L$) has a large margin over the best-performing extraction method (BL*) on all of the test datasets. Secondly, we further compare our method with CopyRNN, and the results indicate that our worst ParaNet$_L$ still achieves better performance than CopyRNN. Note that ConNet does not perform as well as we expect, and is slightly worse than CopyRNN on most datasets. The main reason for this may be that directly concatenating embeddings of two syntactic tags and words introduces much noise into the encoder, such as POS tag of verb.

Finally, we compare our different models. From the results shown in Table 4, we can observe that ParaNet$_T$ is more effective than ParaNet$_L$. This means that, in combining the word and syntactic

[3]https://tartarus.org/martin/PorterStemmer/

Method	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	0.098	0.113	0.202	0.058	0.116	0.043	0.066	0.125	0.211
ConNet	0.041	0.083	0.094	0.184	0.059	0.117	0.041	0.057	0.119	0.203
ParaNet$_L$	0.047	0.097	0.121	0.208	0.063	0.119	0.043	0.068	0.133	0.224
ParaNet$_T$	0.054	0.098	**0.127**	**0.214**	**0.069**	**0.127**	**0.044**	0.069	0.136	**0.228**
ParaNet$_L$+CoAtt	0.053	0.099	0.125	0.206	0.065	0.123	0.042	0.069	0.134	0.226
ParaNet$_T$+CoAtt	**0.060**	**0.103**	0.125	**0.214**	0.068	0.125	**0.044**	**0.071**	**0.137**	**0.228**

Table 5: Comparisons of predicting absent keyphrases on five scientific datasets.

No.	$\lvert P_i\rvert/\lvert O_l\rvert$ (%)	$\lvert P_i^n\rvert/\lvert O_l^n\rvert$ (%)			
		$n=1$	$n=2$	$n=3$	$n\geq 4$
1	5.05-1.57	0	1.93-0.76	16.52-4.97	37.33-9.82
2	9.87-1.23	21.57-2.41	2.77-0.53	1.50-0.21	0.78-0.03
3	4.90-0.68	0	4.47-0.62	14.99-2.45	16.55-0.81
4	5.82-1.43	0	4.23-0.50	21.42-7.47	16.62-0.84
5	1.37-0.04	0.82-0.03	2.33-0.06	0.62-0.01	0.24-0
6	9.82-0.95	8.80-0.98	13.52-1.32	5.33-0.20	1.44+0.03
Total	**36.83 -5.90**	**31.19-3.42**	**29.25-3.79**	**60.38-15.31**	**72.96-11.47**

Table 6: Comparisons of seriousness of the overlapping phrase problem between ParaNet$_T$+CoAtt and CopyRNN.

tag hidden vectors form encoders, the tree-LSTM model performs better than the hyperbolic tangent function. The reason for this may be that the multiple gating functions in tree-LSTM help ParaNet$_T$ to select the useful information from each encoder. In addition, we can observe that coverage attention mechanism can help to gain better performance in generating present keyphrases. Among our proposed models, ParaNet$_T$+CoAtt achieves the best performance on almost all test datasets.

5.5.2 Prediction of Absent Keyphrase

As mentioned in the work (Meng et al., 2017), the Seq2Seq models can predict absent keyphrases. Therefore, we only compare our method with CopyRNN and ConNet, and evaluate the performance within the recall of the top-10 and top-50 results to see how many absent keyphrases can be correctly predicted.

The results are shown in Table 5. As the results show, our worst model (ParaNet$_L$) can correctly predict more absent keyphrases than CopyRNN. The main reason for this may be that the syntactic tags provide more useful information for identifying a part of absent keyphrases which have special syntactic structures. In addition, we note that ConNet is still slightly worse than CopyRNN in predicting absent keyphrases.

Finally, we compare our four different models for generating absent keyphrases. From the results shown in Table 5, we can observe that ParaNet$_T$ can correctly predict more absent keyphrases than ParaNet$_L$ on all test datasets. As the results in the present keyphrase generation, the tree-LSTM model still performs better than the hyperbolic tangent function in the absent keyphrase generation. In addition, we can observe that coverage attention mechanism can help to correctly predict more absent keyphrases. The reason for this may be that the coverage vector can capture long-distance dependencies. This will help to generate the absent keyphrases which are the non-contiguous subsequences of source text. Among our proposed models, ParaNet$_T$+CoAtt perform better than the other three models on most test datasets.

5.5.3 Reduction of Overlapping Phrases

As mentioned in the Section 1, the important motivation for this work is to alleviate generating the overlapping phrases of keyphrases. Table 6 shows the same statistics as Table 1, compared between the best performing model ParaNet$_T$+CoAtt and CopyRNN. From the results, we observe that, compared with CopyRNN, ParaNet$_T$+CoAtt can significantly alleviate the overlapping phrase problem, especially for the sub-phrase problems No.1, No.3 and No.4. For example, the proportion of the keyphrases suffering from the overlapping problem in all keyphrases has dropped from 42.73% to 36.83%. In addition, we investigate the proportion

Method	F1@10	Method	F1@10
TF-IDF	0.270	ParaNet$_L$	0.186
TextRank	0.097	ParaNet$_T$	0.188
SingleRank	0.256	ParaNet$_L$+CoAtt	0.187
CopyRNN	0.164	ParaNet$_T$+CoAtt	0.191

Table 7: Comparisons of different methods on DUC.

of the keyphrases with the length n which suffer from the i-th sub-problem in all keyphrases with the same length, i.e., $|P_i^n|/|O_l^n|$. We observe that this proportion of 3-grams ($n = 3$) reduces most significantly by up to 15.31%.

In addition to the reduction of the overlapping phrases on KP20k dataset, compared with CopyRNN, ParaNet$_T$+CoAtt can highly rank the correctly predicted keyphrases and rank lowly the overlapping phrases of keyphrases. For example, in the sub-problem No.3, ParaNet$_T$+CoAtt can increase the average ranking of correctly predicted keyphrases from 6.50 to 5.95 at top-10 predictions, and decrease the average ranking of sub-phrases of keyphrases from 2.08 to 2.41.

5.5.4 Cross-Domain Testing

CopyRNN and ParaNet are supervised methods, and are trained on a large-scale dataset in specific scientific domain. Similar to the work (Meng et al., 2017), we expect that our supervised method can learn universal language features that are also effective in other corpora. We thus test our method on new type of text, to see whether the method will work when being transferred to a different domain. We use the popular news article dataset: DUC-2001 (Wan and Xiao, 2008) for our experiments, which consists of 308 news articles and 2,488 manually labeled keyphrases.

The results are shown in Table 7. From these results, we can observe that our models generate a certain number of keyphrases in the new domain,. Though the best ParaNet$_T$+CoAtt falls behind the unsupervised algorithms TF-IDF and SingleRank, the worst ParaNet$_L$ significantly outperforms the TextRank and CopyRNN. In addition, we note that the overlapping phrase problem also exists in DUC dataset. In the experiment, ParaNet$_T$+CoAtt can reduce the total proportion of keyphrases suffering from the overlapping phrase problem from 21.96% to 19.13%.

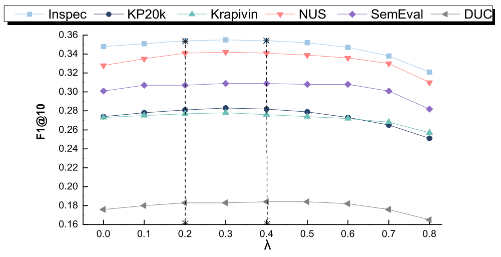

Figure 2: The influence of the weight λ (F1@10).

5.5.5 Influence of Weight Parameter

In this work, we propose the multi-task Seq2Seq network for keyphrase generation, which jointly learns the dominant task of predicting keyphrases and the auxiliary task of predicting POS tags of keyphrases. We employ the weight parameter λ (in Equ. 21) to tune the impacts of the two tasks.

We conduct the experiment to illustrate the influence of the weight parameter λ in ParaNet$_L$, which does not use the coverage attention. The results are shown in Figure 2, in which the F1 at top-10 predictions are given on six datasets. We observe that the performance of ParaNet$_L$ is influenced by changes on the parameter λ. In general, the performance slowly increases and then slowly decreases on six datasets as λ grows. The best-performing settings are $\lambda = 0.5$ on news dataset DUC and $\lambda = 0.3$ on other five scientific datasets, which are finally used to balance two prediction tasks in the comparison experiments.

6 Conclusion

In this study, we propose the parallel Seq2Seq network with the coverage attention to alleviate the overlapping problem (including sub-phrase and super-phrase problems) in existing keyphrase generation methods. In particular, we incorporate the linguistic constraints of keyphrases into the basic Seq2Seq network, and employ multi-task learning framework to enhance generation performance. The experimental results show that the proposed method can significantly outperform the state-of-the-art CopyRNN on scientific datasets, and is also effective in news domain.

References

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Proceedings of EMNLP*, pages 4057–4066.

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

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of AAAI*, pages 1629–1635.

Sujatha Das Gollapalli, Xiao-Li Li, and Peng Yang. 2017. Incorporating expert knowledge into keyphrase extraction. In *Proceedings of AAAI*, 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 ACL*, pages 1631–1640.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of ACL*, pages 1262–1273.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP*, pages 216–223.

Anette Hulth and Beáta B Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of ACL*, pages 537–544.

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.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, pages 1–13.

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*, pages 257–266.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2010. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

Olena Medelyan, Eibe Frank, and Ian H Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of EMNLP*, pages 1318–1327.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of ACL*, pages 582–592.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of EMNLP*, pages 1318–1327.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, pages 317–326.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of ACL*, pages 1073–1083.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of EMNLP*, pages 1526–1534.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, pages 1556–1566.

Yixuan Tang, Weilong Huang, Qi Liu, and Beibei Zhang. 2017. Qalink: Enriching text documents with relevant Q&A site contents. In *Proceedings of CIKM*, pages 3159–3168.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL*, pages 76–85.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of AAAI*, pages 855–860.

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*, pages 254–255.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of EMNLP*, pages 4142–4153.

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*, pages 1349–1358.