Deep Keyphrase Generation

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He[*]**, Peter Brusilovsky, Yu Chi**
School of Computing and Information
University of Pittsburgh
Pittsburgh, PA, 15213
{rui.meng, saz31, shh69, daqing, peterb, yuc73}@pitt.edu

Abstract

Keyphrase provides highly-summative information that can be effectively used for understanding, organizing and retrieving text content. Though previous studies have provided many workable solutions for automated keyphrase extraction, they commonly divided the to-be-summarized content into multiple text chunks, then ranked and selected the most meaningful ones. These approaches could neither identify keyphrases that do not appear in the text, nor capture the real semantic meaning behind the text. We propose a generative model for keyphrase prediction with an encoder-decoder framework, which can effectively overcome the above drawbacks. We name it as *deep keyphrase generation* since it attempts to capture the deep semantic meaning of the content with a deep learning method. Empirical analysis on six datasets demonstrates that our proposed model not only achieves a significant performance boost on extracting keyphrases that appear in the source text, but also can generate absent keyphrases based on the semantic meaning of the text. Code and dataset are available at https://github.com/memray/seq2seq-keyphrase.

1 Introduction

A keyphrase or keyword is a piece of short, summative content that expresses the main semantic meaning of a longer text. The typical use of a keyphrase or keyword is in scientific publications to provide the core information of a paper. We use the term "keyphrase" interchangeably with "keyword" in the rest of this paper, as both terms have an implication that they may contain multiple words. High-quality keyphrases can facilitate the understanding, organizing, and accessing of document content. As a result, many studies have focused on ways of automatically extracting keyphrases from textual content (Liu et al., 2009; Medelyan et al., 2009a; Witten et al., 1999). Due to public accessibility, many scientific publication datasets are often used as test beds for keyphrase extraction algorithms. Therefore, this study also focuses on extracting keyphrases from scientific publications.

Automatically extracting keyphrases from a document is called *keypharase extraction*, and it has been widely used in many applications, such as information retrieval (Jones and Staveley, 1999), text summarization (Zhang et al., 2004), text categorization (Hulth and Megyesi, 2006), and opinion mining (Berend, 2011). Most of the existing keyphrase extraction algorithms have addressed this problem through two steps (Liu et al., 2009; Tomokiyo and Hurst, 2003). The first step is to acquire a list of keyphrase candidates. Researchers have tried to use n-grams or noun phrases with certain part-of-speech patterns for identifying potential candidates (Hulth, 2003; Le et al., 2016; Liu et al., 2010; Wang et al., 2016). The second step is to rank candidates on their importance to the document, either through supervised or unsupervised machine learning methods with a set of manually-defined features (Frank et al., 1999; Liu et al., 2009, 2010; Kelleher and Luz, 2005; Matsuo and Ishizuka, 2004; Mihalcea and Tarau, 2004; Song et al., 2003; Witten et al., 1999).

There are two major drawbacks in the above keyphrase extraction approaches. First, these methods can only extract the keyphrases that ap-

---

[*]Corresponding author

pear in the source text; they fail at predicting meaningful keyphrases with a slightly different sequential order or those that use synonyms. However, authors of scientific publications commonly assign keyphrases based on their semantic meaning, instead of following the written content in the publication. In this paper, we denote phrases that do not match any contiguous subsequence of source text as **absent keyphrases**, and the ones that fully match a part of the text as **present keyphrases**. Table 1 shows the proportion of present and absent keyphrases from the document abstract in four commonly-used datasets, from which we can observe large portions of absent keyphrases in all the datasets. The absent keyphrases cannot be extracted through previous approaches, which further prompts the development of a more powerful keyphrase prediction model.

Second, when ranking phrase candidates, previous approaches often adopted machine learning features such as TF-IDF and PageRank. However, these features only target to detect the importance of each word in the document based on the statistics of word occurrence and co-occurrence, and are unable to reveal the full semantics that underlie the document content.

Table 1: Proportion of the present keyphrases and absent keyphrases in four public datasets

Dataset	# Keyphrase	% Present	% Absent
Inspec	19,275	55.69	44.31
Krapivin	2,461	44.74	52.26
NUS	2,834	67.75	32.25
SemEval	12,296	42.01	57.99

To overcome the limitations of previous studies, we re-examine the process of *keyphrase prediction* with a focus on how real human annotators would assign keyphrases. Given a document, human annotators will first read the text to get a basic understanding of the content, then they try to digest its essential content and summarize it into keyphrases. Their generation of keyphrases relies on an understanding of the content, which may not necessarily use the exact words that occur in the source text. For example, when human annotators see "*Latent Dirichlet Allocation*" in the text, they might write down "*topic modeling*" and/or "*text mining*" as possible keyphrases. In addition to the semantic understanding, human annotators

might also go back and pick up the most important parts, based on syntactic features. For example, the phrases following "*we propose/apply/use*" could be important in the text. As a result, a better keyphrase prediction model should understand the semantic meaning of the content, as well as capture the contextual features.

To effectively capture both the semantic and syntactic features, we use recurrent neural networks (RNN) (Cho et al., 2014; Gers and Schmidhuber, 2001) to compress the semantic information in the given text into a dense vector (i.e., semantic understanding). Furthermore, we incorporate a copying mechanism (Gu et al., 2016) to allow our model to find important parts based on positional information. Thus, our model can generate keyphrases based on an understanding of the text, regardless of the presence or absence of keyphrases in the text; at the same time, it does not lose important in-text information.

The contribution of this paper is three-fold. First, we propose to apply an RNN-based generative model to keyphrase prediction, as well as incorporate a copying mechanism in RNN, which enables the model to successfully predict phrases that rarely occur. Second, this is the first work that concerns the problem of absent keyphrase prediction for scientific publications, and our model recalls up to 20% of absent keyphrases. Third, we conducted a comprehensive comparison against six important baselines on a broad range of datasets, and the results show that our proposed model significantly outperforms existing supervised and unsupervised extraction methods.

In the remainder of this paper, we first review the related work in Section 2. Then, we elaborate upon the proposed model in Section 3. After that, we present the experiment setting in Section 4 and results in Section 5, followed by our discussion in Section 6. Section 7 concludes the paper.

2 Related Work

2.1 Automatic Keyphrase Extraction

A keyphrase provides a succinct and accurate way of describing a subject or a subtopic in a document. A number of extraction algorithms have been proposed, and the process of extracting keyphrases can typically be broken down into two steps.

The first step is to generate a list of phrase can-

didates with heuristic methods. As these candidates are prepared for further filtering, a considerable number of candidates are produced in this step to increase the possibility that most of the correct keyphrases are kept. The primary ways of extracting candidates include retaining word sequences that match certain part-of-speech tag patterns (e.g., nouns, adjectives) (Liu et al., 2011; Wang et al., 2016; Le et al., 2016), and extracting important n-grams or noun phrases (Hulth, 2003; Medelyan et al., 2008).

The second step is to score each candidate phrase for its likelihood of being a keyphrase in the given document. The top-ranked candidates are returned as keyphrases. Both supervised and unsupervised machine learning methods are widely employed here. For supervised methods, this task is solved as a binary classification problem, and various types of learning methods and features have been explored (Frank et al., 1999; Witten et al., 1999; Hulth, 2003; Medelyan et al., 2009b; Lopez and Romary, 2010; Gollapalli and Caragea, 2014). As for unsupervised approaches, primary ideas include finding the central nodes in text graph (Mihalcea and Tarau, 2004; Grineva et al., 2009), detecting representative phrases from topical clusters (Liu et al., 2009, 2010), and so on.

Aside from the commonly adopted two-step process, another two previous studies realized the keyphrase extraction in entirely different ways. Tomokiyo and Hurst (2003) applied two language models to measure the phraseness and informativeness of phrases. Liu et al. (2011) share the most similar ideas to our work. They used a word alignment model, which learns a translation from the documents to the keyphrases. This approach alleviates the problem of vocabulary gaps between source and target to a certain degree. However, this translation model is unable to handle semantic meaning. Additionally, this model was trained with the target of title/summary to enlarge the number of training samples, which may diverge from the real objective of generating keyphrases.

Zhang et al. (2016) proposed a joint-layer recurrent neural network model to extract keyphrases from tweets, which is another application of deep neural networks in the context of keyphrase extraction. However, their work focused on sequence labeling, and is therefore not able to predict absent keyphrases.

2.2 Encoder-Decoder Model

The RNN Encoder-Decoder model (which is also referred as sequence-to-sequence Learning) is an end-to-end approach. It was first introduced by Cho et al. (2014) and Sutskever et al. (2014) to solve translation problems. As it provides a powerful tool for modeling variable-length sequences in an end-to-end fashion, it fits many natural language processing tasks and can rapidly achieve great successes (Rush et al., 2015; Vinyals et al., 2015; Serban et al., 2016).

Different strategies have been explored to improve the performance of the Encoder-Decoder model. The attention mechanism (Bahdanau et al., 2014) is a soft alignment approach that allows the model to automatically locate the relevant input components. In order to make use of the important information in the source text, some studies sought ways to copy certain parts of content from the source text and paste them into the target text (Allamanis et al., 2016; Gu et al., 2016; Zeng et al., 2016). A discrepancy exists between the optimizing objective during training and the metrics during evaluation. A few studies attempted to eliminate this discrepancy by incorporating new training algorithms (Marc'Aurelio Ranzato et al., 2016) or by modifying the optimizing objectives (Shen et al., 2016).

3 Methodology

This section will introduce our proposed deep keyphrase generation method in detail. First, the task of keyphrase generation is defined, followed by an overview of how we apply the RNN Encoder-Decoder model. Details of the framework as well as the copying mechanism will be introduced in Sections 3.3 and 3.4.

3.1 Problem Definition

Given a keyphrase dataset that consists of $\mathbf{N}$ data samples, the i-th data sample $(\mathbf{x^{(i)}}, \mathbf{p^{(i)}})$ contains one source text $\mathbf{x^{(i)}}$, and M_i target keyphrases $\mathbf{p^{(i)}} = (\mathbf{p^{(i,1)}}, \mathbf{p^{(i,2)}}, \ldots, \mathbf{p^{(i,M_i)}})$. Both the source text $\mathbf{x^{(i)}}$ and keyphrase $\mathbf{p^{(i,j)}}$ are sequences of words:

$$\mathbf{x^{(i)}} = x_1^{(i)}, x_2^{(i)}, \ldots, x_{L_{\mathbf{x^i}}}^{(i)}$$

$$\mathbf{p^{(i,j)}} = y_1^{(i,j)}, y_2^{(i,j)}, \ldots, y_{L_{\mathbf{p^{(i,j)}}}}^{(i,j)}$$

$L_{\mathbf{x^{(i)}}}$ and $L_{\mathbf{p^{(i,j)}}}$ denotes the length of word sequence of $\mathbf{x^{(i)}}$ and $\mathbf{p^{(i,j)}}$ respectively.

Each data sample contains one source text sequence and multiple target phrase sequences. To apply the RNN Encoder-Decoder model, the data need to be converted into text-keyphrase pairs that contain only one source sequence and one target sequence. We adopt a simple way, which splits the data sample $(\mathbf{x}^{(i)}, \mathbf{p}^{(i)})$ into M_i pairs: $(\mathbf{x}^{(i)}, \mathbf{p}^{(i,1)}), (\mathbf{x}^{(i)}, \mathbf{p}^{(i,2)}), \ldots, (\mathbf{x}^{(i)}, \mathbf{p}^{(i,M_i)})$. Then the Encoder-Decoder model is ready to be applied to learn the mapping from the source sequence to target sequence. For the purpose of simplicity, $(\mathbf{x}, \mathbf{y})$ is used to denote each data pair in the rest of this section, where $\mathbf{x}$ is the word sequence of a source text and $\mathbf{y}$ is the word sequence of its keyphrase.

3.2 Encoder-Decoder Model

The basic idea of our keyphrase generation model is to compress the content of source text into a hidden representation with an encoder and to generate corresponding keyphrases with the decoder, based on the representation . Both the encoder and decoder are implemented with recurrent neural networks (RNN).

The encoder RNN converts the variable-length input sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$ into a set of hidden representation $\mathbf{h} = (h_1, h_2, \ldots, h_T)$, by iterating the following equations along time t:

$$\mathbf{h}_t = f(x_t, \mathbf{h}_{t-1}) \qquad (1)$$

where f is a non-linear function. We get the context vector $\mathbf{c}$ acting as the representation of the whole input $\mathbf{x}$ through a non-linear function q.

$$\mathbf{c} = q(h_1, h_2, ..., h_T) \qquad (2)$$

The decoder is another RNN; it decompresses the context vector and generates a variable-length sequence $\mathbf{y} = (y_1, y_2, ..., y_{T'})$ word by word, through a conditional language model:

$$\begin{aligned} \mathbf{s}_t &= f(y_{t-1}, \mathbf{s}_{t-1}, \mathbf{c}) \\ p(y_t | y_{1,...,t-1}, \mathbf{x}) &= g(y_{t-1}, \mathbf{s}_t, \mathbf{c}) \end{aligned} \qquad (3)$$

where $\mathbf{s}_t$ is the hidden state of the decoder RNN at time t. The non-linear function g is a softmax classifier, which outputs the probabilities of all the words in the vocabulary. y_t is the predicted word at time t, by taking the word with largest probability after $g(\cdot)$.

The encoder and decoder networks are trained jointly to maximize the conditional probability of the target sequence, given a source sequence. After training, we use the beam search to generate phrases and a max heap is maintained to get the predicted word sequences with the highest probabilities.

3.3 Details of the Encoder and Decoder

A bidirectional gated recurrent unit (GRU) is applied as our encoder to replace the simple recurrent neural network. Previous studies (Bahdanau et al., 2014; Cho et al., 2014) indicate that it can generally provide better performance of language modeling than a simple RNN and a simpler structure than other Long Short-Term Memory networks (Hochreiter and Schmidhuber, 1997). As a result, the above non-linear function f is replaced by the **GRU** function (see in (Cho et al., 2014)).

Another forward GRU is used as the decoder. In addition, an attention mechanism is adopted to improve performance. The attention mechanism was firstly introduced by Bahdanau et al. (2014) to make the model dynamically focus on the important parts in input. The context vector $\mathbf{c}$ is computed as a weighted sum of hidden representation $\mathbf{h} = (h_1, \ldots, h_T)$:

$$\begin{aligned} \mathbf{c}_i &= \sum_{j=1}^{T} \alpha_{ij} h_j \\ \alpha_{ij} &= \frac{\exp(a(s_{i-1}, h_j))}{\sum_{k=1}^{T} \exp(a(s_{i-1}, h_k))} \end{aligned} \qquad (4)$$

where $a(s_{i-1}, h_j)$ is a soft alignment function that measures the similarity between s_{i-1} and h_j; namely, to which degree the inputs around position j and the output at position i match.

3.4 Copying Mechanism

To ensure the quality of learned representation and reduce the size of the vocabulary, typically the RNN model considers a certain number of frequent words (e.g. 30,000 words in (Cho et al., 2014)), but a large amount of long-tail words are simply ignored. Therefore, the RNN is not able to recall any keyphrase that contains out-of-vocabulary words. Actually, important phrases can also be identified by positional and syntactic information in their contexts, even though their exact meanings are not known. The copying mechanism (Gu et al., 2016) is one feasible solution that enables RNN to predict out-of-vocabulary words by selecting appropriate words from the source text.

By incorporating the copying mechanism, the probability of predicting each new word y_t consists of two parts. The first term is the probability of generating the term (see Equation 3) and the second one is the probability of copying it from the source text:

$$p(y_t|y_{1,...,t-1}, \mathbf{x})$$
$$= p_g(y_t|y_{1,...,t-1}, \mathbf{x}) + p_c(y_t|y_{1,...,t-1}, \mathbf{x}) \quad (5)$$

Similar to attention mechanism, the copying mechanism weights the importance of each word in source text with a measure of positional attention. But unlike the generative RNN which predicts the next word from all the words in vocabulary, the copying part $p_c(y_t|y_{1,...,t-1}, \mathbf{x})$ only considers the words in source text. Consequently, on the one hand, the RNN with copying mechanism is able to predict the words that are out of vocabulary but in the source text; on the other hand, the model would potentially give preference to the appearing words, which caters to the fact that most keyphrases tend to appear in the source text.

$$p_c(y_t|y_{1,...,t-1}, \mathbf{x}) = \frac{1}{Z} \sum_{j:x_j=y_t} \exp(\psi_c(x_j)), y \in \chi$$
$$\psi_c(x_j) = \sigma(\mathbf{h}_j^T \mathbf{W}_c)\mathbf{s}_t$$
$$(6)$$

where χ is the set of all of the unique words in the source text $\mathbf{x}$, σ is a non-linear function and $\mathbf{W_c} \in \mathbb{R}$ is a learned parameter matrix. $\mathbf{Z}$ is the sum of all the scores and is used for normalization. Please see (Gu et al., 2016) for more details.

4 Experiment Settings

This section begins by discussing how we designed our evaluation experiments, followed by the description of training and testing datasets. Then, we introduce our evaluation metrics and baselines.

4.1 Training Dataset

There are several publicly-available datasets for evaluating keyphrase generation. The largest one came from Krapivin et al. (2008), which contains 2,304 scientific publications. However, this amount of data is unable to train a robust recurrent neural network model. In fact, there are millions of scientific papers available online, each of which contains the keyphrases that were assigned by their authors. Therefore, we collected a large amount of high-quality scientific metadata in the computer science domain from various online digital libraries, including ACM Digital Library, ScienceDirect, Wiley, and Web of Science etc. (Han et al., 2013; Rui et al., 2016). In total, we obtained a dataset of 567,830 articles, after removing duplicates and overlaps with testing datasets, which is 200 times larger than the one of Krapivin et al. (2008). Note that our model is only trained on 527,830 articles, since 40,000 publications are randomly held out, among which 20,000 articles were used for building a new test dataset **KP20k**. Another 20,000 articles served as the validation dataset to check the convergence of our model, as well as the training dataset for supervised baselines.

4.2 Testing Datasets

For evaluating the proposed model more comprehensively, four widely-adopted scientific publication datasets were used. In addition, since these datasets only contain a few hundred or a few thousand publications, we contribute a new testing dataset **KP20k** with a much larger number of scientific articles. We take the title and abstract as the source text. Each dataset is described in detail below.

- **Inspec** (Hulth, 2003): This dataset provides 2,000 paper abstracts. We adopt the 500 testing papers and their corresponding *uncontrolled keyphrases* for evaluation, and the remaining 1,500 papers are used for training the supervised baseline models.

- **Krapivin** (Krapivin et al., 2008): This dataset provides 2,304 papers with full-text and author-assigned keyphrases. However, the author did not mention how to split testing data, so we selected the first 400 papers in alphabetical order as the testing data, and the remaining papers are used to train the supervised baselines.

- **NUS** (Nguyen and Kan, 2007): We use both *author-assigned* and *reader-assigned* keyphrases and treat all 211 papers as the testing data. Since the NUS dataset did not specifically mention the ways of splitting training and testing data, the results of the supervised baseline models are obtained through a five-fold cross-validation.

- **SemEval-2010** (Kim et al., 2010): 288 articles were collected from the ACM Digital

Library. 100 articles were used for testing and the rest were used for training supervised baselines.

- **KP20k**: We built a new testing dataset that contains the titles, abstracts, and keyphrases of 20,000 scientific articles in computer science. They were randomly selected from our obtained 567,830 articles. Due to the memory limits of implementation, we were not able to train the supervised baselines on the whole training set. Thus we take the 20,000 articles in the validation set to train the supervised baselines. It is worth noting that we also examined their performance by enlarging the training dataset to 50,000 articles, but no significant improvement was observed.

4.3 Implementation Details

In total, there are 2,780,316 $\langle$text, keyphrase$\rangle$ pairs for training, in which text refers to the concatenation of the title and abstract of a publication, and keyphrase indicates an author-assigned keyword. The text pre-processing steps including tokenization, lowercasing and replacing all digits with symbol $\langle$digit$\rangle$ are applied. Two encoder-decoder models are trained, one with only attention mechanism (RNN) and one with both attention and copying mechanism enabled (CopyRNN). For both models, we choose the top 50,000 frequently-occurred words as our vocabulary, the dimension of embedding is set to 150, the dimension of hidden layers is set to 300, and the word embeddings are randomly initialized with uniform distribution in [-0.1,0.1]. Models are optimized using Adam (Kingma and Ba, 2014) with initial learning rate $= 10^{-4}$, gradient clipping $= 0.1$ and dropout rate $= 0.5$. The max depth of beam search is set to 6, and the beam size is set to 200. The training is stopped once convergence is determined on the validation dataset (namely early-stopping, the cross-entropy loss stops dropping for several iterations).

In the generation of keyphrases, we find that the model tends to assign higher probabilities for shorter keyphrases, whereas most keyphrases contain more than two words. To resolve this problem, we apply a simple heuristic by preserving only the first single-word phrase (with the highest generating probability) and removing the rest.

4.4 Baseline Models

Four unsupervised algorithms (Tf-Idf, TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), and ExpandRank (Wan and Xiao, 2008)) and two supervised algorithms (KEA (Witten et al., 1999) and Maui (Medelyan et al., 2009a)) are adopted as baselines. We set up the four unsupervised methods following the optimal settings in (Hasan and Ng, 2010), and the two supervised methods following the default setting as specified in their papers.

4.5 Evaluation Metric

Three evaluation metrics, the macro-averaged *precision*, *recall* and *F-measure* (F_1) are employed for measuring the algorithm's performance. Following the standard definition, precision is defined as the number of correctly-predicted keyphrases over the number of all predicted keyphrases, and recall is computed by the number of correctly-predicted keyphrases over the total number of data records. Note that, when determining the match of two keyphrases, we use Porter Stemmer for pre-processing.

5 Results and Analysis

We conduct an empirical study on three different tasks to evaluate our model.

5.1 Predicting Present Keyphrases

This is the same as the keyphrase extraction task in prior studies, in which we analyze how well our proposed model performs on a commonly-defined task. To make a fair comparison, we only consider the present keyphrases for evaluation in this task. Table 2 provides the performances of the six baseline models, as well as our proposed models (i.e., RNN and CopyRNN). For each method, the table lists its F-measure at top 5 and top 10 predictions on the five datasets. The best scores are highlighted in bold and the underlines indicate the second best performances.

The results show that the four unsupervised models (Tf-idf, TextTank, SingleRank and ExpandRank) have a robust performance across different datasets. The ExpandRank fails to return any result on the KP20k dataset, due to its high time complexity. The measures on NUS and SemEval here are higher than the ones reported in (Hasan and Ng, 2010) and (Kim et al., 2010), probably because we utilized the paper abstract instead of the full text for training, which may

Method	Inspec		Krapivin		NUS		SemEval		KP20k	
	F$_1$@5	F$_1$@10	F$_1$@5	F$_1$@10	F$_1$@5	F$_1$@10	F$_1$@5	F$_1$@10	F$_1$@5	F$_1$@10
Tf-Idf	0.223	<u>0.304</u>	0.113	0.143	0.139	0.181	0.120	<u>0.184</u>	0.105	0.130
TextRank	<u>0.229</u>	0.275	0.172	0.147	0.195	0.190	<u>0.172</u>	0.181	0.180	0.150
SingleRank	0.214	0.297	0.096	0.137	0.145	0.169	0.132	0.169	0.099	0.124
ExpandRank	0.211	0.295	0.096	0.136	0.137	0.162	0.135	0.163	N/A	N/A
Maui	0.040	0.033	<u>0.243</u>	<u>0.208</u>	<u>0.249</u>	<u>0.261</u>	0.045	0.039	<u>0.265</u>	<u>0.227</u>
KEA	0.109	0.129	0.096	0.136	0.068	0.081	0.027	0.027	0.180	0.163
RNN	0.000	0.000	0.002	0.001	0.005	0.004	0.004	0.003	0.138	0.009
CopyRNN	**0.292**	**0.336**	**0.302**	**0.252**	**0.342**	**0.317**	**0.291**	**0.296**	**0.328**	**0.255**

Table 2: The performance of predicting present keyphrases of various models on five benchmark datasets

filter out some noisy information. The performance of the two supervised models (i.e., Maui and KEA) were unstable on some datasets, but Maui achieved the best performances on three datasets among all the baseline models.

As for our proposed keyphrase prediction approaches, the RNN model with the attention mechanism did not perform as well as we expected. It might be because the RNN model is only concerned with finding the hidden semantics behind the text, which may tend to generate keyphrases or words that are too general and may not necessarily refer to the source text. In addition, we observe that 2.5% (70,891/2,780,316) of keyphrases in our dataset contain out-of-vocabulary words, which the RNN model is not able to recall, since the RNN model can only generate results with the 50,000 words in vocabulary. This indicates that a pure generative model may not fit the extraction task, and we need to further link back to the language usage within the source text. The CopyRNN model, by considering more contextual information, significantly outperforms not only the RNN model but also all baselines, exceeding the best baselines by more than 20% on average. This result demonstrates the importance of source text to the extraction task. Besides, nearly 2% of all correct predictions contained out-of-vocabulary words.

The example in Figure 1(a) shows the result of predicted present keyphrases by RNN and Copy-RNN for an article about video search. We see that both models can generate phrases that relate to the topic of information retrieval and video. However most of RNN predictions are high-level terminologies, which are too general to be selected as keyphrases. CopyRNN, on the other hand,

predicts more detailed phrases like "video metadata" and "integrated ranking". An interesting bad case, "rich content" coordinates with a keyphrase "video metadata", and the CopyRNN mistakenly puts it into prediction.

5.2 Predicting Absent Keyphrases

As stated, one important motivation for this work is that we are interested in the proposed model's capability for predicting absent keyphrases based on the "understanding" of content. It is worth noting that such prediction is a very challenging task, and, to the best of our knowledge, no existing methods can handle this task. Therefore, we only provide the RNN and CopyRNN performances in the discussion of the results of this task. Here, we evaluate the performance within the recall of the top 10 and top 50 results, to see how many absent keyphrases can be correctly predicted. We use the absent keyphrases in the testing datasets for evaluation.

Dataset	RNN		CopyRNN	
	R@10	R@50	R@10	R@50
Inspec	0.032	0.057	**0.051**	**0.101**
Krapivin	0.095	0.155	**0.116**	**0.195**
NUS	0.060	0.107	**0.078**	**0.144**
SemEval	0.045	0.065	**0.049**	**0.075**
KP20k	0.076	0.129	**0.115**	**0.189**

Table 3: Absent keyphrases prediction performance of RNN and CopyRNN on five datasets

Table 3 presents the recall results of the top 10/50 predicted keyphrases for our RNN and CopyRNN models, in which we observe that the CopyRNN can, on average, recall around 8%

(15%) of keyphrases at top 10 (50) predictions. This indicates that, to some extent, both models can capture the hidden semantics behind the textual content and make reasonable predictions. In addition, with the advantage of features from the source text, the CopyRNN model also outperforms the RNN model in this condition, though it does not show as much improvement as the present keyphrase extraction task. An example is shown in Figure 1(b), in which we see that two absent keyphrases, "video retrieval" and "video indexing", are correctly recalled by both models. Note that the term "indexing" does not appear in the text, but the models may detect the information "index videos" in the first sentence and paraphrase it to the target phrase. And the CopyRNN successfully predicts another two keyphrases by capturing the detailed information from the text (highlighted text segments).

5.3 Transferring the Model to the News Domain

RNN and CopyRNN are supervised models, and they are trained on data in a specific domain and writing style. However, with sufficient training on a large-scale dataset, we expect the models to be able to learn universal language features that are also effective in other corpora. Thus in this task, we will test our model on another type of text, to see whether the model would work when being transferred to a different environment.

We use the popular news article dataset **DUC-2001** (Wan and Xiao, 2008) for analysis. The dataset consists of 308 news articles and 2,488 manually annotated keyphrases. The result of this analysis is shown in Table 4, from which we could see that the CopyRNN can extract a portion of correct keyphrases from a unfamiliar text. We also report the baseline performance included in (Hasan and Ng, 2010). The performance of CopyRNN is better than TextRank (Mihalcea and Tarau, 2004) and KeyCluster (Liu et al., 2009), but lags behind the other three baselines. It is worth noting that the hyperparameters of baseline models, such as number of recalled keyphrases for Tf-Idf and SingleRank, are carefully tuned and may drastically affect the results. However for CopyRNN we simply report its F_1 score of top 10 predicted phrases ($F_1@10$).

As it is transferred to a corpus in a completely different type and domain, the model encounters more unknown words and has to rely more on the positional and syntactic features within the text. In this experiment, the CopyRNN recalls 766 keyphrases. 14.3% of them contain out-of-vocabulary words, and many names of persons and places are correctly predicted.

Model	F_1	Model	F_1
Tf-Idf	0.270	**ExpandRank**	0.269
TextRank	0.097	**KeyCluster**	0.140
SingleRank	0.256	**CopyRNN**	0.165

Table 4: Keyphrase prediction performance of CopyRNN on DUC-2001. The model is trained on scientific publication and evaluated on news.

6 Discussion

Our experimental results demonstrate that the CopyRNN model not only performs well on predicting present keyphrases, but also has the ability to generate topically relevant keyphrases that are absent in the text. In a broader sense, this model attempts to map a long text (i.e., paper abstract) with representative short text chunks (i.e., keyphrases), which can potentially be applied to improve information retrieval performance by generating high-quality index terms, as well as assisting user browsing by summarizing long documents into short, readable phrases.

Thus far, we have tested our model with scientific publications and news articles, and have demonstrated that our model has the ability to capture universal language patterns and extract key information from unfamiliar texts. We believe that our model has a greater potential to be generalized to other domains and types, like books, online reviews, etc., if it is trained on a larger data corpus. Also, we directly applied our model, which was trained on a publication dataset, into generating keyphrases for news articles without any adaptive training. We believe that with proper training on news data, the model would make further improvement.

Additionally, this work mainly studies the problem of discovering core content from textual materials. Here, the encoder-decoder framework is applied to model language; however, such a framework can also be extended to locate the core information on other data resources, such as summarizing content from images and videos.

Title: Towards content-based **relevance ranking** for **video search**

Abstract: Most existing web **video search** engines index videos by file names, URLs, and surrounding texts. These types of **video metadata** roughly describe the whole video in an abstract level without taking the rich content, such as semantic content descriptions and speech within the video, into consideration. Therefore the relevance ranking of the video search results is not satisfactory as the details of video contents are ignored. In this paper we propose a novel relevance ranking approach for Web-based video search using both **video metadata** and the rich content contained in the videos. To leverage real content into ranking, the videos are segmented into shots, which are smaller and more semantic-meaningful retrievable units, and then more detailed information of video content such as semantic descriptions and speech of each shots are used to improve the retrieval and ranking performance. With **video metadata** and content information of shots, we developed an **integrated ranking** approach, which achieves improved ranking performance. We also introduce machine learning into the ranking system, and compare them with IR-model (information retrieval model) based method. The evaluation results demonstrate the effectiveness of the proposed ranking methods.

(a) Present Keyphrase

RNN: 1. information retrieval; **2. video search**; 3. search engine; 4. video content; 5. machine learning; 6. web video; 7. content based; 8. semantic content; 9. web based video; 10. web based

CopyRNN: 1. information retrieval; **2. video search**; 3. ranking; 4. machine learning; **5. relevance ranking**; **6. video metadata**; **7. integrated ranking**; 8. web video; 9. web video search; 10. rich content

(b) Absent Keyphrase

RNN: **1. video retrieval**; 2. relevance feedback; 3. video summarization; 4.query expansion; **5.video indexing**; 6.semantic web; 7. multimedia retrieval; 8. image retrieval; 9. web search; 10. query processing

CopyRNN: **1. video retrieval**; 2. web search; 3. content ranking; 4. content based retrieval; 5. content retrieval; 6. **video indexing**; 7. relevance feedback; 8. video ranking; 9. semantic web; 10. content based video retrieval; 34. **content based ranking**; 61. **video segmentation**

Figure 1: An example of predicted keyphrase by RNN and CopyRNN. Phrases shown in bold are correct predictions.

7 Conclusions and Future Work

In this paper, we proposed an RNN-based generative model for predicting keyphrases in scientific text. To the best of our knowledge, this is the first application of the encoder-decoder model to a keyphrase prediction task. Our model summarizes phrases based the deep semantic meaning of the text, and is able to handle rarely-occurred phrases by incorporating a copying mechanism. Comprehensive empirical studies demonstrate the effectiveness of our proposed model for generating both present and absent keyphrases for different types of text. Our future work may include the following two directions.

- In this work, we only evaluated the performance of the proposed model by conducting off-line experiments. In the future, we are interested in comparing the model to human annotators and using human judges to evaluate the quality of predicted phrases.

- Our current model does not fully consider correlation among target keyphrases. It would also be interesting to explore the multiple-output optimization aspects of our model.

Erratum

We mistakenly reported the micro-averaged scores for all models instead of macro-averaged ones. We have updated all the scores to macro-averaged. As the difference between micro-averaged and macro-averaged score is marginal, this mistake doesn't affect any conclusions we drew in the submitted version. We sincerely apologize for this mistake and we thank Wang Chen from The Chinese University of Hong Kong for pointing it out.

References

M. Allamanis, H. Peng, and C. Sutton. 2016. A Convolutional Attention Network for Extreme Summarization of Source Code. *ArXiv e-prints* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *IJCNLP*. Citeseer, pages 1162–1170.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-specific keyphrase extraction .

Felix A Gers and E Schmidhuber. 2001. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks* 12(6):1333–1340.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'14, pages 1629–1635. http://dl.acm.org/citation.cfm?id=2892753.2892779.

Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multi-theme documents. In *Proceedings of the 18th International Conference on World Wide Web*. ACM, New York, NY, USA, WWW '09, pages 661–670. https://doi.org/10.1145/1526709.1526798.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* .

Shuguang Han, Daqing He, Jiepu Jiang, and Zhen Yue. 2013. Supporting exploratory people search: a study of factor transparency and user control. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, pages 449–458.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 365–373.

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

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

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

Steve Jones and Mark S Staveley. 1999. Phrasier: a system for interactive document retrieval using keyphrases. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 160–167.

Daniel Kelleher and Saturnino Luz. 2005. Automatic hypertext keyphrase detection. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'05, pages 1608–1609. http://dl.acm.org/citation.cfm?id=1642293.1642576.

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*. Association for Computational Linguistics, pages 21–26.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Mikalai Krapivin, Aliaksandr Autayeu, and Maurizio Marchese. 2008. Large dataset for keyphrases extraction. Technical Report DISI-09-055, DISI, Trento, Italy.

Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. 2016. *Unsupervised Keyphrase Extraction: Introducing New Kinds of Words to Keyphrases*, Springer International Publishing, Cham, pages 665–671.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 135–144.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 366–376.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, pages 257–266.

Patrice Lopez and Laurent Romary. 2010. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Stroudsburg, PA, USA, SemEval '10, pages 248–251. http://dl.acm.org/citation.cfm?id=1859664.1859719.

Sumit Chopra Marc'Aurelio Ranzato, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR, San Juan, Puerto Rico* .

Yutaka Matsuo and Mitsuru Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools* 13(01):157–169.

Olena Medelyan, Eibe Frank, and Ian H Witten. 2009a. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*. Association for Computational Linguistics, pages 1318–1327.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009b. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '09, pages 1318–1327. http://dl.acm.org/citation.cfm?id=1699648.1699678.

Olena Medelyan, Ian H Witten, and David Milne. 2008. Topic indexing with wikipedia. In *Proceedings of the AAAI WikiAI workshop*. volume 1, pages 19–24.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. Association for Computational Linguistics.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International Conference on Asian Digital Libraries*. Springer, pages 317–326.

Meng Rui, Han Shuguang, Huang Yun, He Daqing, and Brusilovsky Peter. 2016. Knowledge-based content linking for online textbooks. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence*. The Institute of Electrical and Electronics Engineers, pages 18–25.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 379–389. http://aclweb.org/anthology/D/D15/D15-1044.pdf.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1683–1692. http://www.aclweb.org/anthology/P16-1159.

Min Song, Il-Yeol Song, and Xiaohua Hu. 2003. Kpspotter: a flexible information gain-based keyphrase extraction system. In *Proceedings of the 5th ACM international workshop on Web information and data management*. ACM, pages 50–53.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Takashi Tomokiyo and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment - Volume 18*. Association for Computational Linguistics, Stroudsburg, PA, USA, MWE '03, pages 33–40. https://doi.org/10.3115/1119282.1119287.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

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

Minmei Wang, Bo Zhao, and Yihua Huang. 2016. *PTR: Phrase-Based Topical Ranking for Automatic Keyphrase Extraction in Scientific Publications*, Springer International Publishing, Cham, pages 120–128.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*. ACM, pages 254–255.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382* .

Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 836–845. https://aclweb.org/anthology/D16-1080.

Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. 2004. World wide web site summarization. *Web Intelligence and Agent Systems: An International Journal* 2(1):39–53.