

Unsupervised Open-domain Keyphrase Generation

Lam Thanh Do^{♣*} Pritom Saha Akash[♡] Kevin Chen-Chuan Chang^{♡*}

[♣]Hanoi University of Science and Technology, Viet Nam

[♡]University of Illinois at Urbana-Champaign, USA

lam.dt183573@sis.hust.edu.vn

{pakash2, kcchang}@illinois.edu

Abstract

In this work, we study the problem of unsupervised open-domain keyphrase generation, where the objective is a keyphrase generation model that can be built without using human-labeled data and can perform consistently across domains. To solve this problem, we propose a seq2seq model that consists of two modules, namely *phraseness* and *informativeness* module, both of which can be built in an unsupervised and open-domain fashion. The phraseness module generates phrases, while the informativeness module guides the generation towards those that represent the core concepts of the text. We thoroughly evaluate our proposed method using eight benchmark datasets from different domains. Results on in-domain datasets show that our approach achieves state-of-the-art results compared with existing unsupervised models, and overall narrows the gap between supervised and unsupervised methods down to about 16%. Furthermore, we demonstrate that our model performs consistently across domains, as it overall surpasses the baselines on out-of-domain datasets.¹.

1 Introduction

Keyphrases are short word sequences that describe the core concepts of the text. The prediction of keyphrases for a text is a task that has received much attention recently. It is a crucial problem as its outputs can be useful for a variety of downstream tasks such as building digital libraries (Gutwin et al., 1999; Witten et al., 2009), document summarization (Litvak and Last, 2008), document visualization (Chuang et al., 2012) and so on.

There are mainly two approaches to keyphrase prediction, namely *keyphrase extraction* (Mihalcea and Tarau, 2004; Florescu and Caragea, 2017a; Bennani-Smires et al., 2018) and *keyphrase generation* (Meng et al., 2017; Chen et al., 2019; Yuan

et al., 2020; Shen et al., 2022). Keyphrase extraction *highlights* keyphrases that appear within the text. On the other hand, keyphrase generation *generates* keyphrases based on the understanding of the given text and therefore allows predicting absent keyphrases alongside present ones (Meng et al., 2017). This ability has made keyphrase generation receive more attention than keyphrase extraction in recent years, as human also tend to use keyphrases that are absent from the text.

Most of the existing keyphrase generation models use manually labeled data for training (Meng et al., 2017; Chen et al., 2018, 2019; Yuan et al., 2020; Ahmad et al., 2021). However, obtaining labeled data is often the most expensive component of any machine learning model, and this is the same for keyphrase generation. Compared to labeled data, access to unlabeled data is easier and mostly available. For example, the arXiv dataset (Clement et al., 2019) containing metadata (e.g., title, abstract) of 1.7 million research articles is readily available on Kaggle. Therefore, it is more desirable to construct a keyphrase generation model in an unsupervised fashion. Furthermore, in practice, the model may have to encounter texts that come from various domains or even unseen ones. Therefore, another attractive property of a keyphrase generation model is the ability to handle open-domain documents.

Considering the above scenario, we propose a new problem called **Unsupervised Open-domain Keyphrase Generation**. Similar to every keyphrase generation methods, the model of our objective is given a text x as input, and as output, it generates a set of keyphrases $\{y\}$. Both x and y are word sequences. Furthermore, the model of our objective should satisfy two requirements: 1) it can be built using only an unlabeled corpus, denoted as D ; 2) it can effectively handle inputs from across domains.

This is a challenging task because we do not

*Work done while visiting Cazoodle Inc.

¹Code and data will be available at <https://github.com/ForwardDataLab/UOKG>.

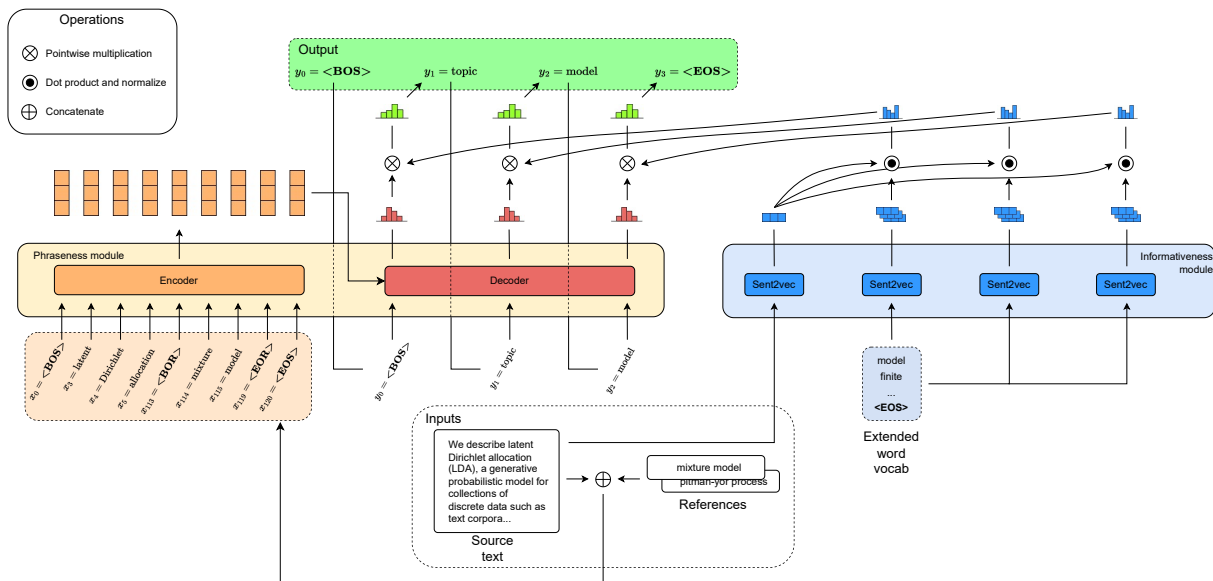


Figure 1: Overview of our proposed model

have access to labeled data from which to learn the patterns for keyphrases. Additionally, we also need our model to work across domains. This is difficult because there might exist different patterns for keyphrases for different domains. None of the existing work addresses these challenges. For instance, supervised keyphrase generation models (Meng et al., 2017; Chen et al., 2018, 2019; Yuan et al., 2020; Ahmad et al., 2021) not only require manually labeled data for training but are also known to perform poorly when being moved out-of-domain. On the other hand, (Shen et al., 2022) propose AutoKeyGen, which uses pseudo-labeled data to train a seq2seq model in a weakly-supervised fashion, thereby removing the need for human annotation effort. However, similar to supervised models, the weakly-supervised approach taken by AutoKeyGen does not enable it to maintain performance in unseen domains.

Therefore, to solve our problem, we propose an unsupervised keyphrase generation model that can work across domains. The **key idea** is to modularize a seq2seq model into two modules. The motivation for modularizing is to decompose keyphrase generation into two simpler problems where each of which can be addressed in an unsupervised and open-domain setting. The first module, named the *phraseness* module, is responsible for generating phrases, while the second module, named the *informativeness* module, guides the generation toward the phrases that represent the most crucial concepts of the text.

The phraseness module is a retrieval-augmented seq2seq model, where the retriever assists the seq2seq component in generating absent phrases alongside present ones. This module can be built in an unsupervised fashion because it leverages noun phrases to index the retriever and to train the seq2seq model, which can easily be obtained using open-sourced software libraries such as NLTK (Bird et al., 2009), and therefore does not require human annotation effort. Furthermore, the phraseness module can also be built in an open-domain fashion, thanks to 1) the part-of-speech information incorporated into the seq2seq model, which allows copying words to form grammatically correct noun phrases regardless of domains; 2) the fact that the retriever can be further indexed with domain-relevant information, to provide reliable references.

The informativeness module is another seq2seq model, where a phrase is likely to be generated if it contains words that are informative to the given text. Inspired by embedding-based unsupervised keyphrase extraction (UKE) methods, we quantify informativeness of a word and a text based on their closeness in meaning, which is measured via the similarity between their embeddings. We choose this method of evaluating informativeness over other UKE methods (e.g. graph-based, statistics based) since it supports not only present phrases, but also absent ones. Similar to the phraseness module, the informativeness module can also be built in an unsupervised and open-domain fashion.

This is obtained by using a domain-general, unsupervised text embedding model (e.g. Sent2Vec (Pagliardini et al., 2018)).

We summarize the contributions of our paper. **Firstly**, we propose a new problem called *unsupervised open-domain keyphrase generation*. **Secondly**, we design a model for solving the problem. Our proposed model is a seq2seq model that consists of two modules, one is responsible for generating phrases and the other guides the generation towards the phrases that represent the core concepts of the text. **Finally**, we conduct extensive experiments on multiple datasets across domains to demonstrate the effectiveness of our model as we contrast it against multiple strong baselines.

2 Proposed method

Figure 1 illustrates our proposed framework. We propose a seq2seq model that consists of two modules, namely *phraseness* and *informativeness* module. We adopt the two terms *phraseness* and *informativeness* from (Tomokiyo and Hurst, 2003), to describe the desirable criteria a keyphrase should satisfy. *Phraseness* refers to the degree to which a word sequence is considered a phrase, and *informativeness* refers to how well the phrase illustrates the core concepts of the text. Each of the two modules guarantees a criterion mentioned above. In particular, the phraseness module generates (present and absent) phrases, while the informativeness module guides the generation toward phrases that describe the core concepts of the text. In the following sections, we will describe in detail the two modules, as well as how they are combined to generate keyphrases.

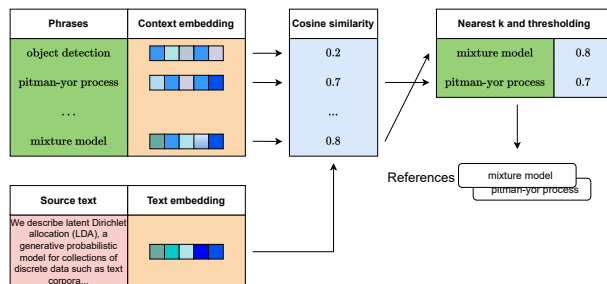


Figure 2: Illustration of the retrieval of references.

2.1 Phraseness module

In order to generate keyphrases, it is crucial to know how to first generate phrases. We emphasize the difference between a keyphrase and a phrase

- the former needs to be informative to the given text, while the latter does not. It has been shown that keyphrases mostly take the form of noun phrases (Chuang et al., 2012). Also, recent work on keyphrase generation has shown that absent keyphrases can often be retrieved from other texts (Ye et al., 2021), suggesting that absent phrases can be found similarly. Therefore, a simple solution to obtaining phrases is to extract noun phrases as present phrases and retrieve related noun phrases as absent ones.

However, this simple solution may not be optimal. Since the retrieved phrases are originally used in other texts, they may not be suitable to describe the concepts of the given text. We demonstrate this limitation using the example in Figure 3a. In this example, the absent phrases obtained via retrieval describe concepts related to “topic modeling”. However, our desired outputs need to also describe concepts related to “author modeling”.

The above problem could be mitigated if we also consider the given text alongside the retrieved noun phrases. In the example above, relevant phrases such as “author topic distributions” can be generated by combining “author”, which is from the given text, and “topic distributions”, which is one of the retrieved phrases. With this in mind, we employ a *retrieval-augmented seq2seq model* as the phraseness module. First, a set of related but absent noun phrases is retrieved, which we will now refer to as *references*. Then, a seq2seq model generates noun phrases based on both the text and the references.

2.1.1 Retriever

Figure 2 describes the retrieval of references given a text. To obtain references for the input, we leverage existing noun phrases observed in other documents. We assume that a noun phrase is related to a text if it occurs in contexts similar to that text. With this in mind, we collect noun phrases from documents in the unlabeled corpus D to form a phrase bank B . We index each noun phrase $z \in B$ with a *context embedding*, denoted as c_z , which is obtained by averaging the embeddings of the documents in which z appears in. We obtain the embeddings of texts by using Sent2Vec (Pagliardini et al., 2018), an unsupervised sentence embedding model. To retrieve references for a text x , we first use Sent2Vec to compute its embedding, denoted as v_x , and then retrieve the top- k phrases z based on the following retrieval score $R_x(z) = \cos(c_z, v_x)$.

<p>Input: We introduce the author-topic model, a generative model for documents that extends Latent Dirichlet Allocation (LDA; Blei, Ng, & Jordan, 2003) to include authorship information. Each author is associated with a multinomial distribution over topics and each topic is associated with a multinomial distribution over words. A document with multiple authors is modeled as a distribution over topics that is a mixture of the distributions associated with the authors. We apply the model to a collection of 1,700 NIPS conference papers and 160,000 CiteSeer abstracts. Exact inference is intractable for these datasets and we use Gibbs sampling to estimate the topic and author distributions. We compare the performance with two other generative models for documents, which are special cases of the author-topic model: LDA (a topic model) and a simple author model in which each author is associated with a distribution over words rather than a distribution over topics. We show topics recovered by the author-topic model, and demonstrate applications to computing similarity between authors and entropy of author output.</p>
<p>Retrieved noun phrases: topic distributions, word distribution, novel topic model, topic models, topic distribution, pitman-yor process, latent topics, hdp, statistical topic models, multinomial distributions, lda model, latent dirichlet allocation model, dirichlet distribution, hierarchical dirichlet process, collapsed gibbs</p>

(a) Example of using retrieval to predict absent phrases

<p>Input: A framework for the analysis of error in global illumination algorithms. In this paper we identify sources of error in global illumination algorithms and derive bounds for each distinct category. Errors arise from three sources: inaccuracies in the boundary data, discretization, and computation. Boundary data consists of surface geometry, reflectance functions, and emission functions, all of which may be perturbed by errors in measurement or simulation, or by simplifications made for computational efficiency. Discretization error is introduced by replacing the continuous radiative transfer equation with a finite-dimensional linear system, usually by means of boundary elements and a corresponding projection method. Finally, computational errors perturb the finite-dimensional linear system through imprecise form factors, inner products, visibility, etc., as well as by halting iterative solvers after a finite number of steps. Using the error taxonomy introduced in the paper we examine existing global illumination algorithms and suggest new avenues of research.</p>
<p>EmbedRank: discretization error, discretization, global illumination algorithms, computational errors, finite-dimensional linear system, computation, iterative solvers, continuous radiative transfer equation, error taxonomy, error</p>
<p>TextRank: error, computational errors, form, projection, linear, functions, boundary, illumination, radiative transfer, computation</p>
<p>Groundtruth: discretization, boundary elements, global illumination, reflectance functions</p>

(b) Example of EmbedRank and TextRank’s prediction, along with the groundtruth keyphrases

Figure 3: Examples

Furthermore, in order to prevent retrieving unreliable references, we filter those whose retrieval scores are smaller than a threshold τ . We denote the set of references for x as Z_x .

As mentioned above, we can further index the retriever with other corpora, denoted as D' , from different domains. To do this, all we need to do is to update the phrase bank B with new phrases from D' and update the context embeddings of every phrase that occur in both D and D' .

2.1.2 Seq2Seq model

Input representation. The seq2seq model takes as inputs not only the source text x but also its references Z_x , to generate phrases. The text and its references are combined into a single input \tilde{x} , defined as

$$\tilde{x} = [\text{BOS}] x [\text{BOR}] Z_x [\text{EOR}] [\text{EOS}] \quad (1)$$

where [BOS] and [EOS] are respectively the beginning and end of sentence token. The two tokens [BOR] and [EOR] signals the start and end of the reference block. In addition, the references are separated by a [SEP] token.

Model architecture. We employ Transformer (Vaswani et al., 2017) with copy mechanism (Gu et al., 2016; See et al., 2017) as the architecture of our seq2seq model. First, the encoder receives the word embeddings of the tokens in \tilde{x} , producing a sequence of encoder hidden states $\mathbf{h} = \{h_i\}_{i=1}^{|\tilde{x}|}$. The decoder takes the embeddings of the previously generated words $\mathbf{y}_{<t}$ and the encoder hidden states, outputting the decoder hidden state s_t . For each input, we build an extended vocabulary $V_{\tilde{x}}$, which is the union of the decoder’s vocabulary V and the words in the augmented input \tilde{x} . Finally, we compute the phraseness probability of predicting a word from $V_{\tilde{x}}$ as $P_{\text{pn}}(y_t | \mathbf{y}_{<t}, \tilde{x}) = p_{\text{gen}} P_{\text{pn}}^V(y_t | \mathbf{y}_{<t}, \tilde{x}) +$

$(1 - p_{\text{gen}}) P_{\text{pn}}^C(y_t | \mathbf{y}_{<t}, \tilde{x})$. Here, $P_{\text{pn}}^V(y_t | \mathbf{y}_{<t}, \tilde{x}) = \text{softmax}(W^V s_t)$ is the distribution over the word vocabulary V , $p_{\text{gen}} = \text{sigmoid}(W_s^g s_t + W_y^g y_{t-1})$ is the soft switch between generating and copy. All the W terms are trainable parameters, and we omit the bias terms for less cluttered notation.

We incorporate part-of-speech information to copy words from \tilde{x} . More formally, $P_{\text{pn}}^C(y_t = w | \mathbf{y}_{<t}, \tilde{x}) = \sum_{\tilde{x}_i=w} a_i^t$, where $a_i^t = \text{softmax}(e_i^t)$, and $e_i^t = \text{FF}_h(\tilde{h}_i^T) \text{FF}_s(\tilde{s}_t)$. Here, $\tilde{h}_i = \text{concat}(h_i, l_{\tilde{x}_i})$ is the encoder hidden state of \tilde{x}_i enhanced with its part-of-speech embedding $l_{\tilde{x}_i}$. Similarly, $\tilde{s}_t = \text{concat}(s_t, l_{y_t})$ is the decoder hidden state enhanced with the part-of-speech embedding of the previously generated word. FF_h and FF_s denotes the feedforward neural networks, whose purposes are to help project \tilde{h}_i and \tilde{s}_t into the same semantic space.

Model training. For every document $x_i \in D$, we maximize $\log P_{\text{pn}}(\mathbf{y} = \mathbf{z} | \tilde{x})$, where

$$P_{\text{pn}}(\mathbf{y} = \mathbf{z} | \tilde{x}) = \prod_{t=1}^T P_{\text{pn}}(y_t = z_t | \mathbf{y}_{<t}, \tilde{x}) \quad (2)$$

for the phrases $\mathbf{z} = \{z_t\}$, which include the present noun phrases and the references. To encourage the model to generate absent phrases instead of just copying it from the references, we randomly mask some references and train the model to generate them.

2.2 Informativeness module

Knowing to generate phrases is not sufficient to obtain keyphrases. It is also important to guide the generation towards the phrases that are informative to the input. Previous work on unsupervised keyphrase extraction offer multiple classes of methods, namely graph-based, statistics-based and embedding-based, for evaluating informativeness

of phrases (for more details, see Section 5). Graph-based and statistics-based methods are not suitable in our setting. These methods utilize only in-text information and therefore cannot determine the informativeness of absent phrases. On the other hand, embedding-based methods evaluate informativeness of a phrase based on its closeness in meaning with the input text. As a result, these methods can support both present and absent phrases. We therefore adopt the idea of embedding-based methods in building our informativeness module.

Let us define $S(\mathbf{a}, \mathbf{b}) = \max(0, v_a^T v_b)$ as the similarity score between two pieces of text, where v_a, v_b are embeddings obtained using Sent2Vec. Using this score, we define the informativeness distribution $P_{\text{in}}(\mathbf{y}|\mathbf{x})$, by decomposing it into conditional distributions of each word given the previous context. More formally, $P_{\text{in}}(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^T P_{\text{in}}(y_t|\mathbf{y}_{<t}, \mathbf{x})$, where

$$P_{\text{in}}(y_t = w|\mathbf{y}_{<t}, \mathbf{x}) \propto \begin{cases} S(w, \mathbf{x}), & \text{if } w \neq [\text{EOS}] \\ S(\mathbf{y}_{<t}, \mathbf{x}), & \text{otherwise} \end{cases} \quad (3)$$

The probability $P_{\text{in}}(y_t = w|\mathbf{y}_{<t}, \mathbf{x})$ is normalized over the extended word vocabulary $V_{\tilde{\mathbf{x}}}$, which is the same one used by the phraseness module. Intuitively, a word has high probability of being generated if that word has close meaning to the text. The [EOS] token is likely to be generated if the currently generated phrase $\mathbf{y}_{<t}$ already form an informative phrase.

2.3 Combining phraseness and informativeness

Generating keyphrases require us to enforce both phraseness and informativeness on the output sequence. A simple solution is to adopt the approaches taken by existing unsupervised keyphrase extraction methods, which enforce the two criteria sequentially. In particular, they either 1) form phrases first, then choose those that are most informative as keyphrases; or 2) choose informative words first, then form keyphrases using these words. However, both approaches may not be optimal. The first approach may include uninformative words in the prediction, while the second rigidly assume that a keyphrase should only contain keywords. We illustrate the limitation of these approaches using an example, shown in Figure 3b. Here, we show the predictions of EmbedRank (Bennani-Smires et al., 2018), which takes approach 1) and TextRank (Mihalcea and Tarau, 2004), which takes approach 2).

Both of them fail to predict the golden keyphrase “global illumination”. EmbedRank redundantly include the word “algorithms”, while TextRank only outputs “illumination”, as “global” is not predicted as a keyword.

This problem could be alleviated if both phraseness and informativeness is considered when forming the keyphrase. In the example above, the word “algorithms” should be excluded, since it neither contributes to the informativeness of the phrase, nor it is required to make the phrase understandable. On the other hand, the word “global” may not be among the most informative words to the text, however, this word is essential as excluding it results in a phrase with a different concept.

In light of this, we propose to generate keyphrases, one word at a time, where each word is generated if it is predicted by both the phraseness and informativeness module. To this end, we propose to combine the two modules in a product-of-experts fashion (Hinton, 2002). In particular, the conditional distribution of a keyphrase given a text is defined as follows

$$\begin{aligned} P_{\text{kp}}(\mathbf{y}|\mathbf{x}) &\propto P_{\text{pn}}(\mathbf{y}|\tilde{\mathbf{x}})^\lambda \cdot P_{\text{in}}(\mathbf{y}|\mathbf{x}) \\ &\propto \prod_{t=1}^T P_{\text{pn}}(y_t|\mathbf{y}_{<t}, \tilde{\mathbf{x}})^\lambda \cdot P_{\text{in}}(y_t|\mathbf{y}_{<t}, \mathbf{x}) \end{aligned} \quad (4)$$

where λ is a hyperparameter for balancing the two modules.

The idea of combining two language models using the product-of-experts has previously been studied for the task of unsupervised abstractive summarization (Zhou and Rush, 2019). To the best of our knowledge, we are the first to use this idea in unsupervised keyphrase generation. In the above paragraphs, we also discussed why it is a suitable choice.

2.4 Keyphrase decoding

To decode keyphrases, we employ beam search to search for keyphrases based on $s(\mathbf{y}) = -\log P_{\text{kp}}(\mathbf{y}|\mathbf{x})$. As beam search tend to favor shorter keyphrases, we employ the length normalization strategy similarly to that described in (Sun et al., 2019), which is to divide $s(\mathbf{y})$ by $|\mathbf{y}| + \alpha$, where α is a length penalty factor.

It has been shown in previous work that positional information is useful for the prediction of present keyphrases (Florescu and Caragea, 2017b; Gallina et al., 2020). Therefore, it is desirable to incorporate this feature into our model. Furthermore,

Dataset name	Language	Type	valid/test docs	#kps/doc	%absent	%overlap
SemEval	English	Scientific	144/100	15.4	58.2	38.8
Inspec	English	Scientific	1500/500	9.7	22.7	40
NUS	English	Scientific	50/161	11.6	53.1	53.3
Krapivin	English	Scientific	1844/460	5.3	50.5	53.5
StackExchange	English	Technical Question	16000/16000	2.7	48.9	45.1
DUC-2001	English	News	50/268	8.1	2.7	14.7
KPTimes	English	News	10000/20000	5	46.2	7.5
OpenKP	English	News	6616/6614	2.2	10.9	12.7

Table 1: Statistics of testing datasets.

we found that the model tends to generate absent keyphrases that are entirely new. This behavior may not be desirable for downstream tasks such as document retrieval, where we need to associate documents with common keyphrases. Based on the above discussion, we propose to rerank the beam search results using the following score

$$\hat{s}(\mathbf{y}) = \frac{s(\mathbf{y})}{|\mathbf{y}| + \alpha} \times b(\mathbf{y}) \quad (5)$$

$$b(\mathbf{y}) = \begin{cases} \beta, & \text{if } \mathbf{y} \text{ is absent and } \mathbf{y} \in B \\ 1, & \text{if } \mathbf{y} \text{ is absent and } \mathbf{y} \notin B \\ \frac{\log_2(1 + \mathcal{P}_x(\mathbf{y}))}{\log_2(1 + \mathcal{P}_x(\mathbf{y})) + 1}, & \text{if } \mathbf{y} \text{ is present} \end{cases} \quad (6)$$

where $b(\mathbf{y})$ is an adjustment weight, β is a hyperparameter for adjusting the scores of absent phrases that exist in the phrase bank B ($\beta < 1$ indicates that we favor $\mathbf{y} \in B$), and $\mathcal{P}_x(\mathbf{y})$ is the word offset position of the phrase \mathbf{y} in the text x . Intuitively, $b(\mathbf{y})$ favors present keyphrases that appear earlier in the text, and absent keyphrases that exist in the phrase bank B .

3 Experiments

3.1 Datasets

We use the documents from the training set of KP20K (Meng et al., 2017) to train our model and to index the retriever in the training phase. It contains the abstracts and titles of 514k scientific articles. In the testing phase, we utilize 8 datasets, namely SemEval (Kim et al., 2013), Inspec (Hulth, 2003), NUS (Nguyen and Kan, 2007), Krapivin (Krapivin et al., 2009), DUC-2001 (Wan and Xiao, 2008), OpenKP (Xiong et al., 2019), StackExchange (Yuan et al., 2020) and KPTimes (Gallina et al., 2019). The title and abstract of an article are concatenated to form a testing document.

The testing datasets are categorized into *in-domain* and *out-of-domain*, by measuring the percentage of keyphrase overlap with the training corpus, i.e. the percentage of golden keyphrases in the testing dataset that also appear in some documents in KP20K. We choose the mean value of ~ 33 as a threshold to classify the testing datasets. As a

result, the in-domain datasets include SemEval, Inspec, NUS, Krapivin and StackExchange, while the other three are out-of-domain.

In the testing phase, besides using KP20K, we also use the training set of StackExchange (300k documents) and KPTimes (260k documents) to further index the phrase bank and the retriever. The purpose of adding these additional sources in the testing phase is to test whether or not our model can easily integrate additional information to work in domains unseen during training, without having it re-trained.

3.2 Baselines & evaluation metrics

Baselines. We adopt five unsupervised keyphrase extraction (UKE) algorithms, namely TF-IDF, TextRank² (Mihalcea and Tarau, 2004), MultiPartiteRank³ (Boudin, 2018), EmbedRank (Bennani-Smires et al., 2018) and Global-Local Rank⁴ (Liang et al., 2021) as baselines.

We also compare our model with AutoKeyGen (Shen et al., 2022), which is the only previous work on unsupervised keyphrase generation. With the permission from the authors, we implemented and report the AutoKeyGen-Copy version. Furthermore, we present CopyRNN (Meng et al., 2017) as a supervised baseline. We employ the Transformer-based pointer-generator network for both AutoKeyGen and CopyRNN, with the same settings as described in A.1. Both AutoKeyGen and CopyRNN are trained using KP20K.

Evaluation metrics. We follow the widely-used strategy and separate the evaluation of present and absent keyphrase generation. We employ macro-average F1 and macro-average Recall for evaluating present and absent keyphrase generation, respectively. We evaluate present keyphrases at top 3 and 5 predictions; and absent keyphrases at top 5 and 10. The predictions as well as the groundtruths are stemmed using Porter Stemmer⁵ (Porter, 1980) and duplicates are removed before evaluation.

3.3 Results

3.3.1 Keyphrase generation for in-domain cases

Table 2 illustrates the performance of our proposed model and the baselines for the five in-domain

²<https://github.com/boudinfl/pke>

³See footnote 1

⁴https://github.com/xnliang98/uke_ccrank

⁵<https://github.com/nltk/nltk/blob/develop/nltk/stem/porter.py>

Present keyphrase generation												
	SemEval		Inspec		NUS		Krapivin		StackExchange		Average	
	F1@3	F1@5	F1@3	F1@5	F1@3	F1@5	F1@3	F1@5	F1@3	F1@5	F1@3	F1@5
TF-IDF	19	23.9	18.7	24.8	22.7	<u>25.9</u>	15.9	15.7	<u>18.8</u>	<u>16.5</u>	19	<u>21.4</u>
TextRank	13.8	17.2	17.7	25	14.9	18.9	11.1	13.3	8.5	8.4	13.2	16.6
MultipartiteRank	18.9	21.4	23.1	26.5	22.7	24.9	19.3	18.5	13.9	13.6	19.6	21
EmbedRank	17.9	21.2	26.2	32.6	17.5	20.8	13.5	15.2	11.8	12.6	17.4	20.5
Global-Local Rank	20.4	<u>23.6</u>	<u>24.5</u>	<u>30.6</u>	22.4	23.7	15	15.2	10.2	9.8	18.5	20.6
AutoKeyGen	16.6 ₄	22.1 ₄	19.4 ₂	23.1 ₃	23.2 ₄	25.7 ₃	19.5 ₇	20.6 ₅	14 ₈	14.9 ₆	18.5 ₃	21.3 ₂
Ours	19.1 ₈	22.2 ₉	19.8 ₈	23.3 ₁₁	26.4₇	27.8₄	22.2₈	21.4₉	27.2₁	25.1₂	23₄	24₄
Supervised - CopyRNN	26.1 ₄	29.7 ₅	19.1 ₅	22.8 ₅	35.5 ₁₃	37.9 ₄	30.3 ₉	30.1 ₆	24.1 ₆	22.4 ₅	27 ₆	28.6 ₃
Absent keyphrase generation												
	SemEval		Inspec		NUS		Krapivin		StackExchange		Average	
	R@5	R@10	R@5	R@10	R@5	R@10	R@5	R@10	R@5	R@10	R@5	R@10
UKE methods	0	0	0	0	0	0	0	0	0	0	0	0
AutoKeyGen	0.7 ₂	1.2 ₃	1.7 ₂	2.8 ₄	1 ₂	1.9 ₅	2.4 ₂	3.8 ₅	1.2 ₂	1.9 ₁	1.4 ₁	2.3 ₂
Ours	1.4₂	2.3₄	2.1₂	3₂	1.8₈	3.1₅	4.5₅	7₂	4.6₁	6.3₂	2.9₂	4.3₂
Supervised - CopyRNN	2.1 ₃	2.7 ₃	3.7 ₃	5.3 ₃	4.4 ₄	6.4 ₇	7.9 ₂	10.7 ₇	2.3 ₂	3.5 ₃	4.1 ₁	5.7 ₃

Table 2: Keyphrase generation performance for in-domain datasets. The best and second-best results for each category are bold and highlighted, respectively. For AutoKeyGen, CopyRNN and our model, we run experiments five times with different random seeds and report the average. The subscript denotes the corresponding standard deviation (e.g. 26.4₇ indicates 26.4±0.7). We report both F1 and Recall in percentage points.

Present keyphrase generation								
	DUC-2001		KPTimes		OpenKP		Average	
	F1@3	F1@5	F1@3	F1@5	F1@3	F1@5	F1@3	F1@5
TF-IDF	8.4	11.3	21.2	21.5	13.5	12.9	14.4	15.2
TextRank	9.8	14	10.1	9.7	9.8	9.3	9.9	11
MultipartiteRank	13.7	18.7	18	18.7	12.7	11.9	<u>14.8</u>	<u>16.4</u>
EmbedRank	20	24.5	10	11.8	6.9	7.3	12.3	14.5
Global-Local Rank	14.6	21.8	10.3	10.7	11.7	10.1	12.2	14.2
AutoKeyGen	7.4 ₆	9.9 ₆	15.9 ₄	16.8 ₃	8.5 ₃	8.9 ₃	10.6 ₃	11.8 ₃
Ours	15.2 ₃	18.2 ₄	20.1 ₁	21.1 ₁	15.9₈	14.2₂	17₃	17.8₂
Supervised - CopyRNN	6.9 ₄	8.5 ₂	19.6 ₇	19.6 ₅	10.4 ₄	10.1 ₃	12.3 ₄	12.7 ₃
Absent keyphrase generation								
	DUC-2001		KPTimes		OpenKP			
	R@5	R@10	R@5	R@10	R@5	R@10		
UKE methods	0	0	0	0	0	0		
AutoKeyGen	-	-	0.2 ₀	0.3 ₀	-	-		
Ours	-	-	3₁	3.6₀	-	-		
Supervised - CopyRNN	-	-	0.2 ₀	0.4 ₁	-	-		

Table 3: Results on out-of-domain datasets.

datasets. We also display the average performance across datasets.

Present keyphrase generation. For predicting present keyphrases, our model is best or second-best on most datasets. On SemEval, our model is slightly inferior to TF-IDF and Global-Local Rank. The results on Inspec are worth noting, as our proposed model is significantly outperformed by UKE methods. This inferior performance may be due to this dataset not favoring generative methods, as even CopyRNN, the supervised baseline, failed to compete with UKE methods on Inspec. This behavior has also been observed in a recent work (Gallina et al., 2020). Although not being able to outperform existing methods on all datasets, our proposed model still achieves the best weighted-average results, outperforming the second-best by about 14% for top 3 predictions and 10% for top 5.

Absent keyphrase generation. For predicting

absent keyphrases, our proposed model outperforms existing work on all datasets. UKE methods cannot be compared with our model, as they only extract present keyphrases. When comparing with AutoKeyGen, we observe that our proposed model have significantly better performance, except for the Inspec dataset where the results are on par. On average, we outperform AutoKeyGen by nearly twice for both top 5 and top 10 predictions.

3.3.2 Keyphrase generation for out-of-domain cases

One important objective of this work is the proposed model’s capability to perform in out-of-domain settings. We show present and absent keyphrase generation performance for out-of-domain datasets in table 3. For absent keyphrase generation, we only report results on KPTimes, as DUC-2001 and OpenKP mainly contain present keyphrases.

Present keyphrase generation. Our model achieves the best or second-best results on all out-of-domain datasets. Similar to the in-domain cases, our model achieves the best weighted-average results despite not being able to outperform all baselines on all datasets. Of the two unsupervised keyphrase generation methods, our proposed model achieves significantly better results than AutoKeyGen in the out-of-domain setting.

Absent keyphrase generation. In the out-of-domain setting. It can be seen that AutoKeyGen fails to generate absent keyphrases, with the re-

In domain	Out of domain
<p>Title: Poset-valued sets or how to build models for linear logics. Abstract: We describe a method for constructing models of linear logic based on the category of sets and relations. The resulting categories are non-degenerate in general; in particular they are not compact closed nor do they have biproducts. The construction is simple, lifting the structure of a poset to the new category. The underlying poset thus controls the structure of this category, and different posets give rise to differently-flavoured models. As a result, this technique allows the construction of models for both, intuitionistic or classical linear logic as desired. A number of well-known models, for example coherence spaces and hypercoherences, are instances of this method.</p>	<p>Title: 'Full horror' of cyclone in southeast Africa yet to emerge: Red Cross Body: MAPUTO/HARARE – Cyclone winds and floods that swept across southeastern Africa affected more than 2.6 million people and could rank as one of the worst weather-related disasters recorded in the southern hemisphere, U.N. officials said on Tuesday. Rescue crews are still struggling to reach victims five days after Cyclone Idai raced in at speeds of up to 170 kph (105 mph) from the Indian Ocean into Mozambique, then its inland neighbors Zimbabwe and Malawi. ...</p>
<p>Present keyphrases</p> <p>Groundtruth: linear logic AutoKeyGen: <u>linear logic</u>, classical linear logic, models, poset, differently-flavoured models Our model: sets, models, <u>linear logics</u>, logics, poset</p>	<p>Present keyphrases</p> <p>Groundtruth: africa, disasters, mozambique, zimbabwe AutoKeyGen: cyclone, <u>africa</u>, satellite images, people, mph Our model: cyclone, floods, <u>africa</u>, <u>mozambique</u>, cyclone winds</p>
<p>Absent keyphrases</p> <p>Groundtruth: categorical models AutoKeyGen: linear models, <u>category models</u>, logic models, general linear models, linear method Our model: linear models, intuitionistic linear logic, relational models, <u>categorical models</u>, kripke models</p>	<p>Absent keyphrases</p> <p>Groundtruth: storms AutoKeyGen: red world, cross world, water told, cyclone people, red cyclone Our model: typhoon, flooding, devastation, <u>storms</u>, tropical cyclone</p>

Figure 4: Two examples of the generated keyphrases from AutoKeyGen and our proposed model. We illustrate the top 5 predictions. Correctly predicted keyphrases are underlined.

call of only 0.3% for top 10 predictions. On the other hand, our model can recall 3.6% of absent keyphrases. This improvement is significant considering that absent keyphrase generation has been pointed out to be a “very challenging task” (Meng et al., 2017).

3.3.3 Comparison to supervised baseline

Although not being able to compete with the supervised baseline on the in-domain datasets, our model has narrowed the gap between supervised and unsupervised keyphrase generation methods. In addition, our model shows remarkable performance on out-of-domain datasets, while the supervised baseline shows poor generalization. It can be seen from Table 2 and 3 that the performance of the supervised baseline plummets on out-of-domain datasets. On the other hand, our model is able to retain performance across domains.

3.4 Ablation study

We perform an ablation study to further understand the role of the components of our proposed model. In particular, we test our model with some components removed, namely the adjustment weight defined in Equation 6, the references and the part-of-speech information. We report the results in Table 4. For KPTimes and OpenKP, we sample 200 and 500 documents from their original validation and test set to perform the ablation study.

We observe that no model in the ablation study

achieve the best performance in all cases. However, the full version of our model shows to be more well-rounded compared to its ablations.

Firstly, the adjustment weight $b(y)$ proves to be crucial, as removing it cause our model’s performance to drop in most cases. This confirms that the positional information is useful in predicting present keyphrases, as has been pointed out by previous work (Florescu and Caragea, 2017b; Gallina et al., 2020). Moreover, prioritizing phrases that exist in the phrase bank also proves to be effective for predicting absent keyphrases. **Next**, removing the references shows to heavily affect absent keyphrase generation, especially on the out-of-domain dataset KPTimes. On the other hand, present keyphrase generation seems not to be affected without using references. **Finally**, the version of our model without part-of-speech information is able to maintain present keyphrase generation performance for the in-domain dataset (Krapivin), but slightly worsens when being moved out-of-domain. For absent keyphrase generation, it seems that part-of-speech information does not help for KPTimes. A possible explanation is that KPTimes mostly contains single-word keyphrases and therefore grammatical information can offer little help in this case.

4 Case study

We display two examples of generated keyphrases from AutoKeyGen and our proposed model in Fig-

Present keyphrase generation				
	Krapivin	DUC-2001	KPTimes	OpenKP
	F1@5	F1@5	F1@5	F1@5
No adjustment weight	17.4	19.3	21.3	10
No references	<u>20.7</u>	<u>18.8</u>	21.4	14.6
No POS	21.1	17.6	21.5	13.1
Full	20.5	18.2	21.8	14
Absent keyphrase generation				
	Krapivin	DUC-2001	KPTimes	OpenKP
	R@10	R@10	R@10	R@10
No adjustment weight	5.8	-	2.5	-
No references	5.5	-	1	-
No POS	<u>7.1</u>	-	3.5	-
Full	7.2	-	<u>3.2</u>	-

Table 4: Ablation study.

ure 4. The first example is from Krapivin, an in-domain dataset, while the second one is from KP-Times, an out-of-domain dataset. For the first example, we observe that both the proposed model and AutoKeyGen correctly predict the groundtruth (present and absent) keyphrases. However, it can be seen that, for generating absent keyphrases, AutoKeyGen only reorders words that are present in the given text. On the other hand, our model can generate keyphrases whose component words are absent, such as “relational models”, “categorical models” and “kripke models”.

In the second example, it is clear that our model predicts more correct keyphrases. We observe that the absent keyphrases generated by AutoKeyGen are highly irrelevant. On the other hand, our model successfully predicts “storms” and also outputs other absent keyphrases that are relevant, although not being within the ground truth keyphrases. This example help shows that our model is better at handling documents from different domains.

5 Related work

5.1 Unsupervised keyphrase extraction

Unsupervised keyphrase extraction (UKE) aims at identifying keyphrases within the text. Currently, there are three main classes of UKE methods, namely statistics-based, graph-based and embedding-based. Statistics-based methods (Campos et al., 2018) employ features such as TF-IDF, word position and casing aspect, to determine the relevance of a candidate phrase.

Graph-based methods typically build a graph from the source text, where a node could be a word or a phrase. Then, different graph-theoretic measures are used to estimate the importance of nodes, and finally phrases are formed based on the top ranked nodes. TextRank (Mihalcea and Tarau,

2004) builds a word graph where a link between two words exists if they co-occur within a window. SingleRank (Wan and Xiao, 2008), CiteTextRank (Gollapalli and Caragea, 2014) employs related documents to better measure similarity between word nodes. TopicRank (Bougouin et al., 2013), Topical PageRank (Liu et al., 2010) incorporate topical information in the graph ranking algorithm. Positional information is used in PositionRank (Florescu and Caragea, 2017a) to favor keyphrases that appear earlier in the text. (Boudin, 2018) utilizes the structure of multi-partite graphs to extract diverse keyphrases.

Embedding-based methods utilize embedding spaces to measure informativeness of candidates. EmbedRank (Bennani-Smires et al., 2018) rank candidates by measuring their distance to the source text in a pretrained sentence embedding space, then an optional diversification step is performed using maximal-marginal relevance to ensure diversity of extracted keyphrases. (Liang et al., 2021) jointly models local and global context of the document when ranking candidates.

5.2 Unsupervised keyphrase generation

Keyphrase generation aims at predicting both present and absent keyphrases for the source text. To our best knowledge, AutoKeyGen (Shen et al., 2022) is currently the only unsupervised keyphrase generation method. AutoKeyGen trains a seq2seq model on automatically generated silver labeled document-keyphrase pairs. The silver keyphrases are both present and absent, where present ones are extracted, and the absent ones are constructed from the words present in the text.

6 Conclusions

In this paper, we propose a new problem called unsupervised open-domain keyphrase generation. We propose a seq2seq model that consists of two modules, one is responsible for generating phrases while the other guides the generation towards phrases that reflect the core concepts of the given text. Our experiments on eight benchmark datasets from multiple domains demonstrate that our model outperforms existing unsupervised methods and narrows the gap between unsupervised and supervised keyphrase generation models. Furthermore, we demonstrate that the proposed model can perform consistently across domains.

Limitations

One limitation of the proposed method is that it does not consider domain-specific information to evaluate informativeness. The phraseness module has access to domain-specific knowledge, which are the phrases that occur in similar contexts, i.e. the references. On the other hand, the informativeness module only employs a domain-general sentence embedding model to measure informativeness of phrases. Therefore, the integration of both domain-specific and domain-general information for the evaluation of informativeness may be worth further investigation.

Another limitation of this work is that we only tested the proposed method on short texts. Therefore, it is uncertain of the proposed framework's performance on long text documents. Handling long texts could be significantly more difficult than short text, as long texts contain much more information (can discuss a variety of topics).

The final limitation of this work is the absence of experiments on using different sentence embedding models to construct the informativeness module. Therefore, it might be useful to explore the impact of different sentence embedding models on keyphrase generation performance. We leave this for future work.

References

- Wasi Ahmad, Xiao Bai, Soomin Lee, and Kai-Wei Chang. 2021. [Select, extract and generate: Neural keyphrase generation with layer-wise coverage attention](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1389–1404, Online. Association for Computational Linguistics.
- Kamil Bennani-Smires, Claudiu Musat, Andreea Hossman, Michael Baeriswyl, and Martin Jaggi. 2018. [Simple unsupervised keyphrase extraction using sentence embeddings](#). In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 221–229, Brussels, Belgium. Association for Computational Linguistics.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc."
- Florian Boudin. 2018. [Unsupervised keyphrase extraction with multipartite graphs](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 667–672, New Orleans, Louisiana. Association for Computational Linguistics.
- Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. [Topicrank: Graph-based topic ranking for keyphrase extraction](#). In *International joint conference on natural language processing (IJCNLP)*, pages 543–551.
- Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. [Yake! collection-independent automatic keyword extractor](#). In *European Conference on Information Retrieval*, pages 806–810. Springer.
- Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. [Keyphrase generation with correlation constraints](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4057–4066, Brussels, Belgium. Association for Computational Linguistics.
- Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019. [-guided encoding for keyphrase generation](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6268–6275.
- Jason Chuang, Christopher D Manning, and Jeffrey Heer. 2012. "without the clutter of unimportant words" descriptive keyphrases for text visualization. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19(3):1–29.
- Colin B. Clement, Matthew Bierbaum, Kevin P. O’Keeffe, and Alexander A. Alemi. 2019. [On the use of arxiv as a dataset](#).
- Corina Florescu and Cornelia Caragea. 2017a. [Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents](#). In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 1105–1115.
- Corina Florescu and Cornelia Caragea. 2017b. [PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115, Vancouver, Canada. Association for Computational Linguistics.
- Ygor Gallina, Florian Boudin, and Beatrice Daille. 2019. [Kptimes: A large-scale dataset for keyphrase generation on news documents](#). *arXiv preprint arXiv:1911.12559*.
- Ygor Gallina, Florian Boudin, and Béatrice Daille. 2020. [Large-scale evaluation of keyphrase extraction models](#). In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, pages 271–278.

- Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28.
- Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. [Incorporating copying mechanism in sequence-to-sequence learning](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.
- Carl Gutwin, Gordon Paynter, Ian Witten, Craig Nevill-Manning, and Eibe Frank. 1999. Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27(1-2):81–104.
- Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.
- Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.
- Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic keyphrase extraction from scientific articles. *Language resources and evaluation*, 47(3):723–742.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction.
- Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. [Unsupervised keyphrase extraction by jointly modeling local and global context](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 155–164, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Marina Litvak and Mark Last. 2008. [Graph-based keyword extraction for single-document summarization](#). In *Coling 2008: Proceedings of the workshop Multi-source Multilingual Information Extraction and Summarization*, pages 17–24, Manchester, UK. Coling 2008 Organizing Committee.
- 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*, pages 366–376.
- Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. [Deep keyphrase generation](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592, Vancouver, Canada. Association for Computational Linguistics.
- Rada Mihalcea and Paul Tarau. 2004. [TextRank: Bringing order into text](#). In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.
- Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, pages 317–326. Springer.
- Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. [Unsupervised learning of sentence embeddings using compositional n-gram features](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.
- Martin F Porter. 1980. An algorithm for suffix stripping. *Program*.
- Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. [Get to the point: Summarization with pointer-generator networks](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.
- Xianjie Shen, Yinghan Wang, Rui Meng, and Jingbo Shang. 2022. [Unsupervised deep keyphrase generation](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11303–11311.
- Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng, and Jian-Yun Nie. 2019. Divgraphpointer: A graph pointer network for extracting diverse keyphrases. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 755–764.
- Takashi Tomokiyo and Matthew Hurst. 2003. [A language model approach to keyphrase extraction](#). In *Proceedings of the ACL 2003 Workshop on Multi-word Expressions: Analysis, Acquisition and Treatment*, pages 33–40, Sapporo, Japan. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Xiaojuan Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860.

Ian H Witten, David Bainbridge, and David M Nichols. 2009. *How to build a digital library*. Morgan Kaufmann.

Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019. Open domain web keyphrase extraction beyond language modeling. *arXiv preprint arXiv:1911.02671*.

Jiacheng Ye, Ruijian Cai, Tao Gui, and Qi Zhang. 2021. [Heterogeneous graph neural networks for keyphrase generation](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2705–2715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. [One size does not fit all: Generating and evaluating variable number of keyphrases](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7961–7975, Online. Association for Computational Linguistics.

Jiawei Zhou and Alexander Rush. 2019. [Simple unsupervised summarization by contextual matching](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5101–5106, Florence, Italy. Association for Computational Linguistics.

A Implementation details

A.1 Phraseness module

Retriever. We extract noun phrases for the documents in the training set of KP20K, StackExchange and KPTime; and form the phrase bank by keeping the noun phrases that occur in at least 5 documents. For obtaining embeddings of documents, we employ the Sent2Vec pretrained model named `sent2vec_wiki_unigrams`⁶, a 600-dimensional sentence embedding model trained on English Wikipedia. We utilize Faiss⁷ (Johnson et al., 2019) for indexing the phrases and their context embeddings.

Seq2seq model. Both the encoder and decoder of the seq2seq model contains 3 layers. The model dimensionality and the word embedding size are both set to 256, and the part-of-speech embedding size is set to 64. The seq2seq model employs attention with 8 heads. The encoder and decoder have separate vocabularies, both contain 40000 words. The encoder and decoder vocabulary contains the frequent words in the unlabeled corpus and among the extracted noun phrases, respectively.

⁶<https://github.com/epfml/sent2vec>

⁷<https://github.com/facebookresearch/faiss>

	1st run	2nd run	3rd run	4th run	5th run
SemEval	-1	-1	-0.5	-1	-1
Inspec	-1	-1	-1	-0.75	-1
NUS	-0.75	-0.75	-0.25	-0.25	-0.5
Krapivin	-0.25	-0.5	-0.25	-0.25	-0.5
StackExchange	1	1	1	1	1
DUC-2001	-1	-1	-1	-1	-1
KPTimes	0.5	0.5	0.75	0.5	1
OpenKP	0.5	-0.25	0.25	-0.25	-0.25

Table 5: Best length penalty values for each dataset in each run.

The seq2seq model is optimized using Adam optimizer (Kingma and Ba, 2014), with a learning rate of 0.0001, gradient clipping = 0.1 and a dropout rate of 0.1. We trained our model in 15 epochs. After every 3 training epoch, the learning rate is reduced by 10%. The seq2seq model contains 34M trainable parameters, and training it for 15 epochs took about 7 hours on a single NVIDIA A40 GPU. We roughly estimate that conducting our experiments, which include training the baseline models, took a total of 150 GPU hours.

A.2 Informativeness module

For the informativeness module, we also employ the pretrained Sent2vec model `sent2vec_wiki_unigrams`, to obtain embeddings of words and texts.

A.3 Keyphrase decoding

We employ beam search with beam size = 100 and beam depth = 6. The balancing hyperparameter λ is set to 0.75. Considering the adjustment weight in Equation 6, we set $\beta = 5/6$ to favor existing phrases in the phrase bank B . For each text, we retrieve 15 references, some of which can be filtered out by the threshold τ , which is set to 0.7.

We use the validation set of each testing dataset to select the value for the length penalty factor α . In particular, we choose $\alpha \in \{-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1\}$ that maximizes the geometric mean of the evaluation metrics, i.e. F1 at top 3 and 5 for present keyphrases and Recall at top 5 and 10 for absent keyphrases. Since the value range of these metrics are different from one another, we divide each metric by its maximum value (that can be found as we try different values of α) for normalization before taking the geometric mean. Table 5 provide the best α values for each dataset in each run in our experiment.