Unsupervised Deep Keyphrase Generation

Xianjie Shen[1], **Yinghan Wang**[2*], **Rui Meng**[3†], **Jingbo Shang**[4‡]
[1,4]University of California, San Diego
[2]Amazon.com Inc
[3]Salesforce Research
{xishen, jshang}@eng.ucsd.edu, yinghanw@amazon.com, ruimeng@salesforce.com,

Abstract

Keyphrase generation aims to summarize long documents with a collection of salient phrases. Deep neural models have demonstrated remarkable success in this task, with the capability of predicting keyphrases that are even absent from a document. However, such abstractiveness is acquired at the expense of a substantial amount of annotated data. In this paper, we present a novel method for keyphrase generation, AutoKeyGen, without the supervision of any annotated doc-keyphrase pairs. Motivated by the observation that an absent keyphrase in a document may appear in other places, in whole or in part, we construct a phrase bank by pooling all phrases extracted from a corpus. With this phrase bank, we assign phrase candidates to new documents by a simple partial matching algorithm, and then we rank these candidates by their relevance to the document from both lexical and semantic perspectives. Moreover, we bootstrap a deep generative model using these top-ranked pseudo keyphrases to produce more absent candidates. Extensive experiments demonstrate that AutoKeyGen outperforms all unsupervised baselines and can even beat a strong supervised method in certain cases.

Introduction

The goal of keyphrase generation is to produce a list of phrases to summarize and characterize a long document (e.g., research papers and news articles). It has a wide spectrum of downstream applications, to name a few, information retrieval (Jones and Staveley 1999), text summarization (Zhang, Zincir-Heywood, and Milios 2004), and text categorization (Hulth and Megyesi 2006).

The trade-off between the capability of generating absent keyphrases (i.e., phrases do not appear in the original document) and the reliance on document-keyphrase supervision has long existed among keyphrase generation methods. Meng et al. (2017) have shown that in scientific documents, up to 50% of keyphrases are absent from the source text, yet they can be helpful for downstream applications such as search and recommendation (Boudin and Gallina 2021). With the advance of deep neural networks, recent studies (Meng et al. 2017; Chen et al. 2019; Sun et al. 2019; Alzaidy, Caragea, and Giles 2019; Yuan et al. 2018; Meng et al. 2019, 2020) are capable of generating keyphrases, according to their semantic relevance to a document, no matter they are present or not. Although these methods have achieved state-of-the-art performance, they require a tremendous number of document-keyphrase pairs to supervise the training, and the data is typically expensive and laborious to collect. For example, Meng et al. (2017) utilized half a million scientific papers with author-annotated keyphrases to train a Sequence-to-Sequence model. Similarly, Xiong et al. (2019) collected 68,000 web pages and have them annotated by professional annotators. While many extractive methods (Hasan and Ng 2010; Shang et al. 2018; Bennani-Smires et al. 2018) do not need any direct supervision and demonstrate robust performance across various datasets, they can only predict phrases that appear in the original document. There are several extractive studies that try to recall absent phrases by including related documents as additional inputs (Wan and Xiao 2008; Florescu and Caragea 2017), however, compared with supervised generative methods, their performance is not competitive.

In this paper, we aim to alleviate this trade-off by proposing an unsupervised method that can generate both present and absent keyphrases without utilizing any human annotation. We observe that absent keyphrases of a document can be present in other documents. Also, some absent keyphrases actually appear in the original document in part (as separate tokens). For example, in the test split of Inspec dataset, a popular benchmark dataset for keyphrase generation, bvcc02

Motivated by these observations, we propose a novel unsupervised deep keyphrase generation method AutoKeyGen as illustrated in Figure 1. Specifically, we first follow previous works (Hasan and Ng 2010; Shang et al. 2018; Bennani-Smires et al. 2018) to extract candidate present keyphrases from all documents and pool them together into a phrase bank. From this present phrase bank, we can draw candidate absent keyphrases for each document through a partial matching process, requiring each stemmed word in the candidate phrase must exist in the input document. To rank both types of keyphrases, we fuse two types of similarity: the TF-IDF score at the lexical level and the embedding similarity at the semantic level. We further utilize these top-ranked present and absent candidates as “silver” data to train a deep generative model. This generative model is expected to augment

[*]Work completed before joining Amazon.
[†]The research was accomplished at University of Pittsburgh.
[‡]Jingbo Shang is the corresponding author.

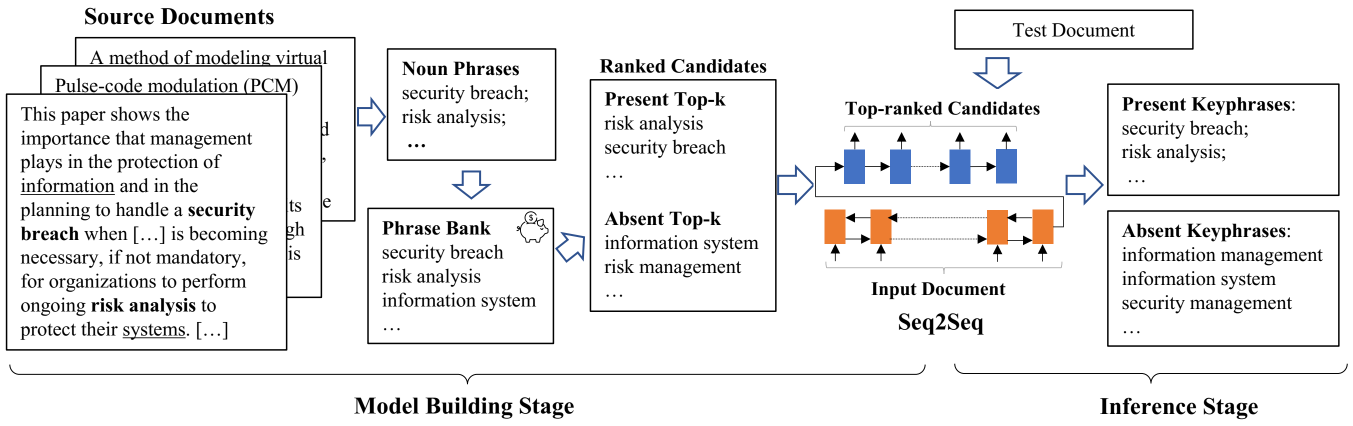

Figure 1: An overview of our proposed AutoKeyGen framework with a real example from NUS dataset.

absent keyphrases with a biased beam search method, which encourages the model to predict words from the input document more than from the vocabulary.

Extensive experiments show that AutoKeyGen consistently outperforms several unsupervised baselines and even beat a strong supervised method (Meng et al. 2017) in certain cases.

Our contributions are summarized as follows:

- We make two important observations about absent keyphrases, demonstrating the feasibility of training abstractive keyphrase generation models in an unsupervised manner.
- We propose a novel unsupervised deep keyphrase generation method AutoKeyGen that can perform well on predicting both present and absent keyphrases.
- Experiment results on five benchmark datasets show the superiority of our method AutoKeyGen over multiple unsupervised baselines. On some datasets, AutoKeyGen even yields better results than a strong supervised method.

Reproducibility. Codes and datasets for reproducing the results of this study will be released on GitHub[1]

Problem Formulation

In this work, we aim to build a keyphrase generation model solely based on a collection of documents $\mathcal{D}$, without any human annotated document-keyphrase labels.

Keyphrase generation is typically formulated and evaluated as a ranking problem. Given an (unseen) input document $\mathbf{x}$, the goal of this task is to output a ranked list of keyphrases $\mathcal{Y}$. We denote an input document as a sequence of tokens, i.e., $\mathbf{x} = [x_1, \ldots, x_{|\mathbf{x}|}]$, where $|\mathbf{x}|$ is the total number of tokens in this document. Depending on whether a keyphrase appears in the input document or not *as a whole unit*, one can categorize the keyphrases in $\mathcal{Y}$ into two ranked lists: (1) present keyphrase ranked list, $\mathcal{Y}^P = [\mathbf{y}_1^p, \ldots, \mathbf{y}_{|\mathcal{Y}^P|}^p]$ and (2) absent keyphrase ranked list: $\mathcal{Y}^A = [\mathbf{y}_1^a, \ldots, \mathbf{y}_{|\mathcal{Y}^A|}^a]$. Here, $|\mathcal{Y}^P|$ and $|\mathcal{Y}^A|$ are the numbers of present and absent keyphrases, respectively. That is, $\mathcal{Y} =< \mathcal{Y}^P, \mathcal{Y}^A >$. Each keyphrase is also a sequence of tokens, which can contain single or multiple tokens.

[1]`https://github.com/Jayshen0/Unsupervised-Deep-Keyphrase-Generation`

Our AutoKeyGen Method

Overview. As shown in Figure 1, the training process of AutoKeyGen consists of three steps: (1) pool present keyphrases from all documents as a phrase bank, and then draw candidate absent keyphrases for each document; (2) for each document, rank all candidates according to the TF-IDF score and the embedding similarity between a document and a candidate phrase; (3) train a Seq2Seq generative model using silver labels (the top-ranked candidates as derived from the second step) to generate final phrases.

When it comes to the inference for new documents, AutoKeyGen will generate candidates using the Seq2Seq model. The output phrases can be absent in the document or missed in the previous candidate list.

Phrase Bank for Absent Keyphrases

Here, we focus on how to construct the phrase bank from a corpus and how to generate absent keyphrase candidates for a specific document.

Phrase Bank Construction. As aforementioned, absent keyphrases in one document would possibly appear in other documents as present keyphrases. For example, in the Inspec dataset, one common benchmark dataset in keyphrase generation, 99% absent keyphrases are present keyphrases in some other documents. Therefore, we first construct a phrase bank by pooling together the present candidates extracted from every document in the raw document collection $\mathcal{D}$. Specifically, we follow the literature (Hasan and Ng 2010; Shang et al. 2018; Bennani-Smires et al. 2018) and extract noun phrases from all documents (using regular expressions to match POS tags with NLTK (Bird 2006)).

Absent Candidate Generation. In many cases, tokens of an absent keyphrase can be found in the source document but not in a verbatim manner. For example, in Inspec, 56.8% of absent keyphrases have all their tokens separately appeared in the input document. This inspires us to utilize a partial matching method to retrieve relevant but absent phrases of a document. Specifically, given an input document $\mathbf{x}$, one can iterate through all the phrases in the phrase bank and take as candidates the phrases whose tokens all appear in $\mathbf{x}$ (after stemming). We enforce the strict requirement of *all tokens* as the phrase bank is huge and there would be too many candidates that can partially appear in $\mathbf{x}$. For the sake of efficiency, we implement this process via an inverted index, mapping document tokens to the phrase bank, so practically we do not have to scan the entire phrase bank for each document.

Ranking Module

A keyphrase generation model is expected to output a ranked list of phrases, so we need to rank the obtained candidates by their importance to the input document. From the literature, we notice that both lexical- and semantic-level metrics are indicative of phrase qualities. Thus, we consider both types of similarity metric for ranking phrases.

Embedding Similarity. According to Bennani-Smires et al. (2018), modern embedding methods, such as Doc2Vec (Lau and Baldwin 2016), are capable of encoding phrases and documents into a shared latent space, then the semantic relatedness can be measured by the cosine similarity in this space. We use a pre-trained Doc2Vec model on the large English Wikipedia corpus and further pre-train it on KP20k, to generate 300-dimension vectors for both an input document and its candidate phrases. Specifically, we denote the embedding of a document $\mathbf{x}$ and a candidate phrase c as $E(\mathbf{x})$ and $E(c)$, respectively. Their semantic similarity is defined as

$$\text{Semantic}(\mathbf{x}, c) = \frac{||E(\mathbf{x}) \cdot E(c)||}{||E(\mathbf{x})|| \cdot ||E(c)||}$$

TF-IDF Score. TF-IDF, measuring the lexical-level similarity, has been observed as a simple yet strong baseline in the literature (Meng et al. 2017; Campos et al. 2018). Specifically, for a document $\mathbf{x}$ in corpus $\mathcal{D}$, the TF-IDF score of phrase c is computed as:

$$\text{Lexical}(\mathbf{x}, c) = \frac{\text{TF}(c, \mathbf{x})}{|\mathbf{x}|} log \frac{|\mathcal{D}|}{\text{DF}(c, \mathcal{D})}$$

where $|\mathbf{x}|$ is the number of word in document $\mathbf{x}$, $\text{TF}(c, \mathbf{x})$ is the term frequency of c in $\mathbf{x}$, $\text{DF}(c, \mathcal{D})$ is the document frequency of c in $\mathcal{D}$.

Fused Ranking. We observe that the embedding-based similarity and TF-IDF behave differently when the input documents are of different lengths. Semantic representations such as Doc2Vec are more reliable for short and medium documents (Lau and Baldwin 2016). TF-IDF can perform robustly when a document is considerably long, such as containing more than 1000 words. Therefore, it is intuitive to unify these two metrics for scoring present keyphrases. We propose to combine them using a geometric mean as follows:

$$\text{RankScore}(\mathbf{x}, c) = \sqrt{\text{Semantic}(\mathbf{x}, c)^{\lambda} \cdot \text{Lexical}(\mathbf{x}, c)}$$

The higher the $\text{RankScore}(\mathbf{x}, c)$ is, the more likely a candidate phrase c is considered more important to its document $\mathbf{x}$. λ is a parameter controlling the ratio of contribution between the semantic score and lexical score. The optimal λ can be manually calculated so that the standard deviation of two score distributions are the same. In our experiments, λ is calculated as 1.4 based on KP20K training split.

To demonstrate the effectiveness of our ranking module, we have provided some ablation studies in the Appendix.

Generation Module

With the abundance of our phrase bank, in our experiments, on average, we are able to cover more than 90% of the present keyphrases, however, less than 30% of the absent keyphrases are included in the test split of KP20k dataset. To increase the coverage of absent candidates, we propose to train a Seq2Seq generative model using the highest scored document-keyphrase pairs from the ranking module. Specifically, we pair each document with the top-5 present candidates and top-5 absent candidates, and use these pairs as silver labels for training.

Classical Encoder-Decoder Model. The encoder is implemented with a BiLSTM (Gers and Schmidhuber 2001) and the decoder is an LSTM. The encoder maps a sequence of tokens in $\mathbf{x}$ to a sequence of continuous hidden representations $(\mathbf{h}^1_{enc}, \ldots, \mathbf{h}^{|\mathbf{x}|}_{enc})$ where $|\mathbf{x}|$ is length of the document, an RNN decoder then generates the target keyphrase $(y^1, y^2, \ldots, y^{|y|})$ token-by-token in an auto-regressive manner ($|y|$ denotes the number of tokens in the keyphrase):

$$\begin{aligned} \mathbf{h}^t_{enc} &= f_{enc}(\mathbf{h}^{t-1}_{enc}, x^t) \\ \mathbf{c} &= q(h^1_{enc}, h^2_{enc}, ..., h^{|\mathbf{x}|}_{enc}) \\ \mathbf{h}^{t'}_{dec} &= f_{dec}(\mathbf{h}^{t'-1}_{dec}, o^{t'-1}, \mathbf{c}) \end{aligned}$$

where $\mathbf{h}^t_{enc}$, and $\mathbf{h}^{t'}_{dec}$ are hidden states at time t/t' for encoder and decoder respectively; f_{enc} and f_{dec} are auto-regressive functions implemented by LSTM cells; $o^{t'-1}$ is the predicted output of decoder at time $t'-1$; and $\mathbf{c}$ is the context vector derived from all the hidden states of encoder through a non-linear function q.

At timestep t, the prediction of $y^{t'}$ is determined based on a distribution over a fixed vocabulary, conditioned on the source representations $\mathbf{h}_{enc}$ and previously generated tokens represented as $\mathbf{h}^{t'-1}_{dec}$:

$$p_g(y^{t'}|y^{1,\ldots,t'-1}, \mathbf{x}) = f_{out}(y^{t'-1}, \mathbf{h}^{t'}_{dec}, \mathbf{c})$$

where f_{out} is a non-linear function, typically a softmax classifier with an attention mechanism, that outputs the probabilities over all the words in a preset vocabulary $\mathcal{V}$.

Tailored Generative Decoding. We use a guided beam search to generate diverse keyphrases for a document. Previous work (Meng et al. 2017) has shown that even when gold labels are available, a vanilla Seq2Seq model could collapse and fail to generate high-quality candidate phrases. Since we only train the model with silver labels, to improve the quality of generated phrases, we encourage the decoder model to generate words that appear in the input document $\mathbf{x}$. More specifically, we double the probabilities of the words in $\mathbf{x}$.

Dataset	Train	Valid	Test
KP20k	514,154	19,992	19,987
Inspec	-	1,500	500
Krapivin	-	1,844	460
NUS	-	-	211
SemEval	-	144	100

Table 1: Statistics of datasets. Only the supervised model CopyRNN uses document-keyphrase labels and the validation set. All other methods use raw documents from the KP20k training set as input.

Note that, words that do not appear in the input document can still be generated so the diversity can be preserved. This also matches our observation that many tokens of absent keyphrases can be found in the input document.

Relationship to Copy Mechanism. In fact, our tailored Seq2Seq model reassembles the copy mechanism proposed in (Meng et al. 2017) and can be viewed as a special version by assuming all tokens in a document follow a similar distribution as estimated by the encoder-decoder model.

As shown in Meng et al. (2017), the copy mechanism is useful for generating keywords because it tends to assign high probabilities to the words that exist in the input document. This is achieved by an extra probability term:

$$p_c(y^t|y^{1,\dots,t-1}, \mathbf{x}) = \frac{1}{Z} \sum_{j:x_j=y^t} exp(\psi(\mathbf{x}_j)), y^t \in \mathbf{x}$$
$$\psi(\mathbf{x}_j) = \sigma((\mathbf{h}^j_{dec})^T W)s^t,$$

where σ is a non-linear function, W is a learned parameter matrix, and Z is a normalization term as the sum of predicted scores of words in the document. For CopyRNN, the probability of generating y^t is the sum of p_g and p_c.

Experiments

In this section, we first introduce datasets used in this study, followed by baselines, evaluation metrics, and details of implementation. Then, we present and discuss the experiment results of present keyphrase and absent keyphrase generation.

Experimental Setup

Datasets We follow previous keyphrase generation studies (Meng et al. 2017; Ye and Wang 2018; Meng et al. 2019; Chen et al. 2019) and adopt five scientific publication datasets for evaluation. **KP20k** is the largest dataset in scientific keyphrase studies thus far. There are four other widely-used scientific datsets for comparing different models: **Inspec** (Tomokiyo and Hurst 2003), **Krapivin** (Krapivin, Autaeu, and Marchese 2009), **NUS** (Nguyen and Kan 2007), and **SemEval-2010** (Kim et al. 2010). Table 1 presents the details of all datasets[2].

All the models in our experiments are built on the KP20k training set. Only the supervised model CopyRNN uses document-keyphrase labels and the validation set. All other methods only use raw documents from the KP20k training set. We then apply the model to all five test sets for evaluation.

Baselines Given the unsupervised setting, for a fair comparison, we mainly compare AutoKeyGen with the following 6 unsupervised methods.

- **TF-IDF** (Jones 1972) ranks the extracted noun phrase candidates by term frequency and inverse document frequency in the corpus.
- **TextRank** (Mihalcea and Tarau 2004) uses PageRank algorithm to score phrases.
- **SingleRank** (Wan and Xiao 2008) is an extension of TextRank that uses the number of co-occurrences to weigh edges in the graph. Then the graph-based ranking algorithm is applied to score phrases.
- **ExpandRank** (Wan and Xiao 2008) proposes to use a small number of nearest neighbor documents to provide more information. Due to its efficiency issue, We fail to obtain the outputs on KP20K within a considerable amount of time.
- **EmbedRank** (Bennani-Smires et al. 2018) directly uses embedding similarities to rank present candidate keyphrase and uses Maximal Marginal Relevance (MMR) (Carbinell and Goldstein 2017) to increase the diversity of extracted keyphrases.

For ablation studies, we compare some variants of our **AutoKeyGen** method as follows:

- **AutoKeyGen-OnlyBank** only uses the partial match between the phrase bank and the input document to extract keyphrase candidates without any seq2seq model.
- **AutoKeyGen-OnlyEmbed** ranks the candidate phrases with only the embedding similarity without the TF-IDF information.
- **AutoKeyGen-CopyRNN** utilizes the CopyRNN in place of our tailored decoding.

We also include **Supervised-CopyRNN** (Meng et al. 2017) as a strong supervised method to explore the gap between supervised and unsupervised methods. It is trained with *annotated document-keyphrase pairs* of the KP20K dataset. Please note that all unsupervised methods, including AutoKeyGen, have no access to these annotated pairs. Therefore, we regard it as a kind of *upper bound* of all unsupervised methods.

Evaluation Metrics Following the literature, we evaluate the model performance on generating present and absent keyphrases separately. If a model generates both present/absent keyphrases in a unified ranked list, we split them into two ranked lists by checking whether or not the phrases appear in the input document. The relative ranking between the phrases of the same type is therefore preserved.

We use $R@k$, $F_1@k$, and $F_1@\mathcal{O}$ (Yuan et al. 2018) as main evaluation metrics. Specifically, $F_1@5$, $F_1@10$, and $F_1@\mathcal{O}$ are utilized for evaluating present keyphrases and $R@10$ and $R@20$ for absent keyphrases. We report the macro-average scores over all documents in each test set.

Specifically, given a ranked list of keyphrases, either present or absent, $\hat{\mathcal{Y}} = (\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_{|\hat{\mathcal{Y}}|})$ and the corresponding ground truth keyphrase set $\mathcal{Y}$, we first truncate it with a cut-

[2]Dataset release is from `https://github.com/memray/OpenNMT-kpg-release`

	Kp20K			Inspec			Krapivin			NUS			SemEval		
Model	**@5**	**@10**	**@$\mathcal{O}$**	**@5**	**@10**	**@$\mathcal{O}$**	**@5**	**@10**	**@$\mathcal{O}$**	**@5**	**@10**	**@$\mathcal{O}$**	**@5**	**@10**	**@$\mathcal{O}$**
TF-IDF	7.2	9.4	6.3	24.2	28.0	24.8	11.5	14.0	13.3	11.6	14.2	12.5	16.1	16.7	15.3
SingleRank	9.9	12.4	10.3	21.4	29.7	22.8	9.6	13.6	13.4	13.7	16.2	18.9	13.2	16.9	14.7
TextRank	18.1	15.1	14.1	26.3	27.9	26.0	14.8	13.9	13.0	18.7	19.5	19.9	16.8	18.3	18.1
ExpandRank	N/A	N/A	N/A	21.1	29.5	26.8	9.6	13.6	11.9	13.7	16.2	15.7	13.5	16.3	14.4
EmbedRank	15.5	15.6	15.8	29.5	34.4	32.8	13.1	13.8	13.9	10.3	13.4	14.7	10.8	14.5	13.9
AutoKeyGen	**23.4**	**24.6**	**23.8**	30.3	34.5	**33.1**	**17.1**	15.5	**15.8**	**21.8**	23.3	**23.7**	**18.7**	**24.0**	**22.7**
AutoKeyGen-OnlyBank	22.9	23.1	23.1	29.7	32.8	32.1	15.9	14.3	14.2	20.7	21.8	22.3	16.3	20.9	20.4
AutoKeyGen-OnlyEmbed	21.2	22.9	21.8	29.7	**34.8**	32.7	15.9	**16.4**	14.3	20.4	21.3	22.6	15.3	16.5	15.9
AutoKeyGen-CopyRNN	22.7	24.2	**23.8**	**30.5**	33.2	32.7	16.6	15.1	14.7	21.6	**22.4**	22.7	**18.7**	22.3	21.4
Supervised-CopyRNN	33.1	27.9	35.3	28.5	32.5	33.7	32.0	27.0	35.5	40.2	35.9	43.4	32.9	34.6	35.2

Table 2: F_1 scores of present keyphrase prediction on five scientific publication datasets. Supervised-CopyRNN results (Meng et al. 2020) are listed only to explore the gap between unsupervised methods and supervised methods. The best/2nd-best scores among unsupervised methods are in bold/underlined.

	Kp20K		Inspec		Krapivin		NUS		SemEval	
Model	**R@10**	**R@20**	**R@10**	**R@20**	**R@10**	**R@20**	**R@10**	**R@20**	**R@10**	**R@20**
Other Unsupervised Methods	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ExpandRank	N/A	N/A	0.02	0.05	0.01	0.015	0.005	0.04	0	0.004
AutoKeyGen	**2.3**	**2.5**	**1.7**	**2.1**	**3.3**	**5.4**	**2.4**	**3.2**	**1.0**	**1.1**
AutoKeyGen-OnlyBank	1.8	2.2	1.5	1.7	3.1	4.1	2.1	2.6	0.7	0.9
AutoKeyGen-OnlyEmbed	1.9	2.3	1.4	1.8	3.0	4.5	2.1	2.7	0.9	0.9
AutoKeyGen-CopyRNN	1.8	2.0	1.6	1.9	3.1	4.7	1.9	2.8	**1.0**	**1.1**
Supervised-CopyRNN	11.5	14.0	5.1	6.8	11.6	14.2	7.8	10.0	4.9	5.7

Table 3: Recall scores of absent keyphrase prediction on five scientific datasets. Supervised-CopyRNN results are listed only to explore the gap between unsupervised methods and supervised methods. The best/2nd-best scores among unsupervised methods are in bold/underlined.

off k (i.e., $\hat{\mathcal{Y}}_{:k} = (\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{\min(k,|\hat{\mathcal{Y}}|)})$) and then evaluate its precision and recall:

$$P@k = \frac{|\hat{\mathcal{Y}}_{:k} \cap \mathcal{Y}|}{|\hat{\mathcal{Y}}_{:k}|}, R@k = \frac{|\hat{\mathcal{Y}}_{:k} \cap \mathcal{Y}|}{|\mathcal{Y}|}$$

$F_1@k$ is the harmonic mean of $P@k$ and $R@k$. $F_1@\mathcal{O}$ can be viewed as a special case of $F_1@k$ when $k = |\mathcal{Y}|$. In other words, we only examine the same amount of keyphrases as the ground truth.

We apply Porter Stemmer provided by NLTK (Bird, Klein, and Loper 2009) to both ground truth and predicted keyphrases to determine whether phrases appear in the original document and whether two keyphrases match or not.

Implementation Details For all keyphrase extraction baselines, we utilize the open-source toolkit pke[3] and EmbedRank[4]. The window size of the graph-based models (SingleRank, TextRank, and ExpandRank) has been searched from 2 to 10 and the best performance is reported. For EmbedRank we use default hyperparameter (β=0.55).

The vocabulary $\mathcal{V}$ of Seq2Seq models consists of 50,000 most frequent uncased words. We train the models for 500,000 steps and take the last checkpoint for inference. The dimension of LSTM cell is 256, the embedding dimension is 200, and the max length of the source text is 512. Models are optimized using Adagrad (Duchi, Hazan, and Singer 2011) with the initial learning rate to be 0.001, and it will be linearly decayed by a ratio of 0.8 for every 5 epochs. For inference, the width of beam search is set to 20.

Experimental Results

Results of Present Keyphrase The results of present keyphrase generation are listed in Table 2. Overall, AutoKeyGen achieves the best $F_1@5$, $F_1@10$ and $F_1@\mathcal{O}$ scores among all the unsupervised methods. EmbedRank is arguably the strongest baseline method, however, AutoKeyGen outperforms it on all datasets with a significant margin.

Among AutoKeyGen and its variants, we can see that AutoKeyGen performs the best in most cases. Comparing to AutoKeyGen-OnlyEmbed, the advantage indicates that the TF-IDF can provide additional information to the embedding-based ranking. The gap between them is less noticeable on Inspec dataset. A possible reason is that the average document length of Inspec is the shortest among all, and TF-IDF can performs more robustly when the input text is long. The advantage of AutoKeyGen over AutoKeyGen-OnlyBank

[3]https://github.com/boudinfl/pke

[4]https://github.com/swisscom/ai-research-keyphrase-extraction

Input Document	This paper shows the importance that management plays in the protection of information and in the planning to handle a **security breach** when a **theft of information** happens. Recent thefts of information that have hit major companies have caused concern. These thefts were caused by companies' inability to determine risks associated with the protection of their data and these companies lack of planning to properly manage a **security breach** when it occurs. It is becoming necessary, if not mandatory, for organizations to perform ongoing **risk analysis** to protect their systems. Organizations need to realize that the **theft of information** is a **management issue** as well as a technology one, and that these recent security breaches were mainly caused by business decisions by management and not a lack of technology.
Present	**Ground Truth**: {security breach, risk analysis, management issue, theft of information} **AutoKeyGen** (ordered): security breach, risk analysis, information, security, business decisions, management issue
Absent	**Ground Truth**: {Information security, information system, case of information theft, information security management, human factor, data protection procedure, security management} **AutoKeyGen** (ordered): security risk, information system, information management, information security management, import concern, data mine, security management, data management

Figure 2: A case study of AutoKeyGen from NUS test set. In the input document, present keyphrases are in bold and tokens related to absent keyphrases are underlined. Correctly predicted keyphrases are highlighted in red. The green text indicates correct phrases predicted by the generative module, which is omitted by the noun phrase extractor.

demonstrates that our generation module does generate a certain number of "novel" present phrases in addition to the noun phrase extractor. AutoKeyGen can also perform better or comparably to AutoKeyGen-CopyRNN, suggesting that preferring words in the text brings better present keyphrases.

It is worth mentioning that on the Inspec dataset, our unsupervised AutoKeyGen can even perform better than the Supervised-CopyRNN method.

Results of Absent Keyphrase Table 3 presents the models' performance on absent keyphrase generation. Following (Meng et al. 2017), only the recall score is reported for comparison. Except for ExpandRank, all unsupervised baseline methods are not capable of generating any absent keyphrases, and we refer to them together as "*Other Unsupervised Methods*". Among all unsupervised models, AutoKeyGen achieves the best recall on all the datasets. Though we see that Supervised-CopyRNN outperforms rest models by a large margin, we argue that AutoKeyGen demonstrates a promising way to derive absent keyphrases under the unsupervised setting.

Please note that generating absent keyphrases is challenging even for supervised methods. If we check the relative performance improvement of AutoKeyGen over AutoKeyGen-OnlyBank, the improvement is actually between 15% - 45%, which should be considered as significant.

Case Studies Figure 2 presents an example from the NUS test set, part of which is also shown in the overview of AutoKeyGen, i.e., Figure 1.

As for the absent keyphrase, our method successfully predicts "*information system*" and "*information security management*" with the help of the phrase bank. These two phrases are extracted from other documents. And they are treated as absent candidates since all their tokens appear in this document (after stemming).

Our method not only selects absent keyphrases from the phrase bank, but can also generate keyphrases from the

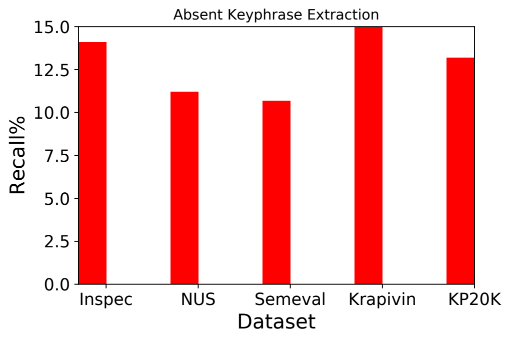

Figure 3: The recall of absent keyphrases using all the phrases in phrase bank on five datasets.

Seq2Seq generative module. In this case, "*security risk*", "*information management*", and "*security management*" are all generated by the generation module. Although some of the predictions do not perfectly match ground truth keyphrases, they convey relevant information about the document, which shows that our model has the potential to produce reliable absent keyphrases.

Coverage of Phrase Bank Figure 3 shows the intersection between the phrase bank and the ground truth absent keyphrases as a set, which is equivalent to the recall of all predictions (recall@inf). It can be seen as the upper-bound absent performance with the extractive part of AutoKeyGen, though such an upper bound can be very hard to reach, since the number of absent candidates in the phrase bank is large. We can see that the phrase bank can cover more than 10% of absent keyphrases on all datasets. The result shows the great potential for unsupervised keyphrase generation, and also calls for better ranking methods for absent keyphrases. It is noteworthy that, many absent phrases are missing in our phrase bank due to the deficiency of our noun phrase extraction method. Better phrase bank construction is worth further investigation.

Discussion on Pseudo-labels In AutoKeyGen, we use top-ranked present/absent phrase candidates as pseudo-labels to train the generative model. Intuitively, present phrases are more reliable than absent ones, however, the absent ones empower the model with the abstractive capability. These thoughts have been confirmed by empirical results as follows: if only absent keyphrases are used for training, the generative model will be misled — average F1@O score on present keyphrases drops by 1.72; if only present keyphrases are used, the performance on absent keyphrases will be lower — average R@10 drops by 0.27.

Related Work

In this section, we mainly review the literature related to the following three areas, (1) keyphrase extraction and generation, (2) word and document embeddings, and (3) encoder-decoder models.

Keyphrase Extraction and Generation

Most of the existing algorithms have addressed the task of keyphrase extraction through two steps (Liu et al. 2009; Tomokiyo and Hurst 2003). The first step is to acquire a list of keyphrase candidates. Previous studies use n-grams or noun phrases with certain part-of-speech patterns to identify potential candidates (Hulth 2003; Le, Nguyen, and Shimazu 2016; Wang, Zhao, and Huang 2016). AutoPhrase (Shang et al. 2018) serves as another option to extract high-quality candidates, using a distant supervised phrase mining method leveraging open-domain knowledge such as Wikipedia. The second step is to rank candidates on their importance to the document using supervised or unsupervised approaches with manually-defined features (Kelleher and Luz 2005; Florescu and Caragea 2017). Florescu and Caragea (2017) tries to score the candidate phrases as the aggregation of its words score, but over-generation errors will happen. Saxena, Mangal, and Jain (2020) transforms keyphrase extraction into classification problem using evolutionary game theory.

The major common drawback of these keyphrase extraction methods is that they can only extract keyphrases that already appear in the source text, but many relevant keyphrases may not appear in the text as consecutive spans. To address this issue, some studies propose to leverage language generation techniques to predict keyphrases, such as CopyRNN (Meng et al. 2017) and CopyCNN (Zhang, Fang, and Weidong 2017). These methods utilize an encoder-decoder architecture, treating the title and main text body as the source information and keyphrases as the target to predict. However, those approaches ignore the leading role of the title in the document structure. To fully leverage the title information, Ye and Wang (2018) propose a semi-supervised learning approach that generates more training pairs and Chen et al. (2019) propose to take title features as a query to guide the decoding process. Swaminathan et al. (2020) apply GAN to keyphrase extraction problem, and it presents a new promising direction for keyphrase problem.

Our work lies in the category of keyphrase generation, but it is significantly different from existing generation studies since our method does not rely on any human annotated document-keyphrase pairs for training.

Word and Document Embeddings

Embeddings (Mikolov et al. 2013) represents words as vectors in a continuous vector space. It's widely used in many NLP problems since embeddings methods take advantage of the classic bag-of-words representation considering it can capture semantic relatedness with acceptable dimensions. The state-of-the-art embeddings methods such as (Lau and Baldwin 2016) is able to infer a vector of a document via an embedding network. In this way, the embeddings of a short phrase and a long document can be represented in a shared vector space, which makes it feasible to derive their semantic relatedness directly with the embedding similarity.

Encoder-Decoder Model

The RNN-based encoder-decoder architecture was first introduced by Cho et al. (2014) and Sutskever, Vinyals, and Le (2014) for machine translation problems. It has also achieved great successes in many other NLP tasks (Serban et al. 2016; Liu, Sands-Meyer, and Audran 2019). The encoder-decoder model is also used for keyphrase extraction problems. Some work (Chen et al. 2020; Allamanis, Peng, and Sutton 2016) tried to copy certain parts of source text when generating the output. See, Liu, and Manning (2017) enhanced this architecture with a pointer-generator network, which allows models to copy words from the source text. Celikyilmaz et al. (2018) proposed an abstractive system where multiple encoders represent the document together with a hierarchical attention mechanism for decoding.

Conclusions and Future Work

In this paper, we propose an unsupervised method for keyphrase generation. Without relying on any human annotation, we derive possible present and absent keyphrases from the corpus. And we combine ranking and generative methods to predict high-quality keyphrases. Extensive experiments demonstrate the superiority of our method over existing unsupervised models on predicting both present and absent keyphrases.

There are still some limitations of our work. In the future, we plan to enhance the quality of the silver label data for training generative models, in order to improve the performance of absent keyphrase generation. One possible way is to filter candidate phrases according to the correlation among keyphrases. Another promising direction is to leverage the intrinsic article structure, such as title-body relations, for self-supervised learning.

Acknowledgements

Our work is sponsored in part by National Science Foundation Convergence Accelerator under award OIA-2040727 as well as generous gifts from Google, Adobe, and Teradata. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes not withstanding any copyright annotation hereon.

References

Allamanis, M.; Peng, H.; and Sutton, C. 2016. A Convolutional Attention Network for Extreme Summarization of Source Code. *CoRR*, abs/1602.03001.

Alzaidy, R.; Caragea, C.; and Giles, C. L. 2019. Bi-LSTM-CRF sequence labeling for keyphrase extraction from scholarly documents. In *The world wide web conference*, 2551–2557.

Bennani-Smires, K.; Musat, C.; Hossmann, A.; Baeriswyl, M.; and Jaggi, M. 2018. Simple Unsupervised Keyphrase Extraction using Sentence Embeddings. In Korhonen, A.; and Titov, I., eds., *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, October 31 - November 1, 2018*, 221–229. Association for Computational Linguistics.

Bird, S. 2006. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, 69–72. Sydney, Australia: Association for Computational Linguistics.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Boudin, F.; and Gallina, Y. 2021. Redefining Absent Keyphrases and their Effect on Retrieval Effectiveness. *arXiv preprint arXiv:2103.12440*.

Campos, R.; Mangaravite, V.; Pasquali, A.; Jorge, A. M.; Nunes, C.; and Jatowt, A. 2018. A Text Feature Based Automatic Keyword Extraction Method for Single Documents. In Pasi, G.; Piwowarski, B.; Azzopardi, L.; and Hanbury, A., eds., *Advances in Information Retrieval - 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings*, volume 10772 of *Lecture Notes in Computer Science*, 684–691. Springer.

Carbinell, J.; and Goldstein, J. 2017. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. *SIGIR Forum*, 51(2): 209–210.

Celikyilmaz, A.; Bosselut, A.; He, X.; and Choi, Y. 2018. Deep Communicating Agents for Abstractive Summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1662–1675. New Orleans, Louisiana: Association for Computational Linguistics.

Chen, W.; Chan, H. P.; Li, P.; and King, I. 2020. Exclusive Hierarchical Decoding for Deep Keyphrase Generation. In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault, J. R., eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, 1095–1105. Association for Computational Linguistics.

Chen, W.; Gao, Y.; Zhang, J.; King, I.; and Lyu, M. R. 2019. Title-Guided Encoding for Keyphrase Generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6268–6275.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Doha, Qatar: Association for Computational Linguistics.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul): 2121–2159.

Florescu, C.; and Caragea, C. 2017. 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)*, 1105–1115. Vancouver, Canada: Association for Computational Linguistics.

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

Hasan, K. S.; and Ng, V. 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*, 365–373. Association for Computational Linguistics.

Hulth, A. 2003. Improved Automatic Keyword Extraction given More Linguistic Knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, 216–223. USA: Association for Computational Linguistics.

Hulth, A.; and Megyesi, B. B. 2006. A Study on Automatically Extracted Keywords in Text Categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 537–544. Sydney, Australia: Association for Computational Linguistics.

Jones, K. S. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28: 11–21.

Jones, S.; and Staveley, M. S. 1999. Phrasier: a system for interactive document retrieval using keyphrases. In *SIGIR '99*.

Kelleher, D.; and Luz, S. 2005. Automatic Hypertext Keyphrase Detection. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, 1608–1609. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kim, S. N.; Medelyan, O.; Kan, M.-Y.; and Baldwin, T. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, 21–26.

Krapivin, M.; Autaeu, A.; and Marchese, M. 2009. Large Dataset for Keyphrases Extraction. In *Departmental Technical Report, University of Trento*. University of Trento.

Lau, J. H.; and Baldwin, T. 2016. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. In Blunsom, P.; Cho, K.; Cohen, S. B.; Grefenstette, E.; Hermann, K. M.; Rimell, L.; Weston, J.; and Yih, S. W., eds., *Proceedings of the 1st Workshop on Representation Learning for NLP, Rep4NLP@ACL 2016, Berlin, Germany, August 11, 2016*, 78–86. Association for Computational Linguistics.

Le, T. T. N.; Nguyen, M. L.; and Shimazu, A. 2016. Unsupervised Keyphrase Extraction: Introducing New Kinds of Words to Keyphrases. *29th Australasian Joint Conference, Hobart, TAS, Australia, December 5-8, 2016.*

Liu, C.; Sands-Meyer, S.; and Audran, J. 2019. The effectiveness of the student response system (SRS) in English grammar learning in a flipped English as a foreign language (EFL) class. *Interact. Learn. Environ.*, 27(8): 1178–1191.

Liu, Z.; Li, P.; Zheng, Y.; and Sun, M. 2009. Clustering to Find Exemplar Terms for Keyphrase Extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 257–266. Singapore: Association for Computational Linguistics.

Meng, R.; Yuan, X.; Wang, T.; Brusilovsky, P.; Trischler, A.; and He, D. 2019. Does order matter? an empirical study on generating multiple keyphrases as a sequence. *arXiv preprint arXiv:1909.03590.*

Meng, R.; Yuan, X.; Wang, T.; Zhao, S.; Trischler, A.; and He, D. 2020. An Empirical Study on Neural Keyphrase Generation. *CoRR*, abs/2009.10229.

Meng, R.; Zhao, S.; Han, S.; He, D.; Brusilovsky, P.; and Chi, Y. 2017. Deep Keyphrase Generation. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).*

Mihalcea, R.; and Tarau, P. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 404–411.

Mikolov, T.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space.

Nguyen, T. D.; and Kan, M.-Y. 2007. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, 317–326. Springer.

Saxena, A.; Mangal, M.; and Jain, G. 2020. KeyGames: A Game Theoretic Approach to Automatic Keyphrase Extraction. In Scott, D.; Bel, N.; and Zong, C., eds., *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, 2037–2048. International Committee on Computational Linguistics.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get To The Point: Summarization with Pointer-Generator Networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).*

Serban, I. V.; Sordoni, A.; Bengio, Y.; Courville, A. C.; and Pineau, J. 2016. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In Schuurmans, D.; and Wellman, M. P., eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, 3776–3784. AAAI Press.

Shang, J.; Liu, J.; Jiang, M.; Ren, X.; Voss, C. R.; and Han, J. 2018. Automated Phrase Mining from Massive Text Corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10): 1825–1837.

Sun, Z.; Tang, J.; Du, P.; Deng, Z.-H.; and Nie, J.-Y. 2019. DivGraphPointer: A Graph Pointer Network for Extracting Diverse Keyphrases. *arXiv preprint arXiv:1905.07689.*

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS.*

Swaminathan, A.; Zhang, H.; Mahata, D.; Gosangi, R.; Shah, R. R.; and Stent, A. 2020. A Preliminary Exploration of GANs for Keyphrase Generation. In Webber, B.; Cohn, T.; He, Y.; and Liu, Y., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, 8021–8030. Association for Computational Linguistics.

Tomokiyo, T.; and Hurst, M. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment*, 33–40.

Wan, X.; and Xiao, J. 2008. Single Document Keyphrase Extraction Using Neighborhood Knowledge. In Fox, D.; and Gomes, C. P., eds., *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 855–860. AAAI Press.

Wang, M.; Zhao, B.; and Huang, Y. 2016. PTR: Phrase-Based Topical Ranking for Automatic Keyphrase Extraction in Scientific Publications. *ICONIP 2016.*

Xiong, L.; Hu, C.; Xiong, C.; Campos, D.; and Overwijk, A. 2019. Open Domain Web Keyphrase Extraction Beyond Language Modeling. *arXiv preprint arXiv:1911.02671.*

Ye, H.; and Wang, L. 2018. Semi-Supervised Learning for Neural Keyphrase Generation. In *EMNLP*, 4142–4153. Association for Computational Linguistics.

Yuan, X.; Wang, T.; Meng, R.; Thaker, K.; Brusilovsky, P.; He, D.; and Trischler, A. 2018. One Size Does Not Fit All: Generating and Evaluating Variable Number of Keyphrases. arXiv:1810.05241.

Zhang, Y.; Fang, Y.; and Weidong, X. 2017. Deep keyphrase generation with a convolutional sequence to sequence model. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, 1477–1485.

Zhang, Y.; Zincir-Heywood, N.; and Milios, E. 2004. World Wide Web Site Summarization. *Web Intelli. and Agent Sys.*, 2(1): 39–53.