Structure-Augmented Keyphrase Generation

Jihyuk Kim
Yonsei University
jihyukkim@yonsei.ac.kr

Myeongho Jeong
Yonsei University
wag9611@yonsei.ac.kr

Seungtaek Choi
Yonsei University
hist0613@yonsei.ac.kr

Seung-won Hwang*
Seoul National University
seungwonh@snu.ac.kr

Abstract

This paper studies the keyphrase generation (**KG**) task for scenarios where structure plays an important role. For example, a scientific publication consists of a short title and a long body, where the title can be used for de-emphasizing unimportant details in the body. Similarly, for short social media posts (*e.g.*, tweets), scarce context can be augmented from titles, though often missing. Our contribution is generating/augmenting structure then encoding these information, using existing keyphrases of other documents, complementing missing/incomplete titles. Specifically, we first extend the given document with related but absent keyphrases from existing keyphrases, to augment missing contexts (*generating structure*), and then, build a graph of keyphrases and the given document, to obtain structure-aware representation of the augmented text (*encoding structure*). Our empirical results validate that our proposed structure augmentation and structure-aware encoding can improve KG for both scenarios, outperforming the state-of-the-art[1].

1 Introduction

Keyphrases not only help human readers to gain immediate insights about a given document, but also make documents queryable, *e.g.*, by hashtagging social posts with predicted keyphrases. Among keyphrase tasks, we focus on tasks that allow words that are not present in the given document, which often be called **absent** words, to be keyphrases, as the input document can be often context-scarce: Many potentially relevant keyphrases are missing in the given document, since the keyphrase can be expressed in terms different from those present in the document (*i.e.*, vocabulary mismatch).

Our goal is to generate keyphrases that are likely to be words absent from the document, especially in the following scenarios: (a) scientific publications with diverse vocabularies, or (b) short social posts, where we found that, from public and real-life datasets we used in our evaluation, 37% of keyphrases in scientific publications, 67% of hashtags, on average, in social media posts are absent in the given document. Thus, we formulate our problem as keyphrase generation (KG) (Meng et al., 2017) of predicting keyphrases, including absent words as well as present words, adopting encoder-decoder architecture.

Field	Text
Title	Methods to tell if a **question** can be **answered** from a paragraph
Question	... Is there any existing research in telling whether or not a **question** is **answerable** ? ... ... I considered textual entailment, but it doesn't seem to be exactly what I'm looking for ...
Existing Keyphrases	..., "**natural language processing**", "formal languages", ...
Gold Keyphrases	"**question answering**", "**natural language processing**"

Table 1: An example of a social Q&A post consisting of a title and the main body question: We present gold keyphrases labeled by the user, and existing keyphrases labeled for other related posts. **Bold-face** words can indicate the topic of the post.

A recent trend is leveraging document structure for KG (Chen et al., 2019b), where metadata in documents, *e.g.*, document title, clarifies the meaning of documents (Kim et al., 2021). For illustration, Table 1 shows a social Q&A post consisting of title (the first row) and main body question (the second row), where we need to predict gold keyphrases "question answering" and "natural language processing" (the last row). The concise title enables readers to focus on important parts of the main question (**bold-face** words), while ignoring details. However, titles often exclude meaningful keyphrases (*e.g.*, "natural language processing" in Table 1) due to their inherent length limitation, and

*Corresponding author.
[1]Our code is available at https://github.com/jihyukkim-nlp/StrAugKG.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 2657–2667
November 7–11, 2021.

making matters worse, titles may not exist at all in some social media posts (*e.g.*, tweets).

Our contribution is to construct a structured document $\mathcal{X}^+$, even when the structure is not given, by leveraging observed keyphrases from other documents in the training dataset, to improve both encoding and decoding. For encoding, keyphrases can be used to emphasize relevant parts in $\mathcal{X}$, similarly done with the title. For decoding, the keyphrases can augment vocabulary, *e.g.*, "natural language processing", from which the decoder can copy words. We stress that our work is designed to work for both closed and open set scenarios, represented by social and scientific document scenario respectively, with the following distinctions:

- Closed set: Hashtags in social media posts are frequently reused (*i.e.*, keyphrases can be copied from observed keyphrases from the training set for decoding).
- Open set: A significant amount of keyphrases of scientific publications (about 20% in our target scenario) have never been observed in the training dataset (*i.e.*, keyphrases candidate set is open-ended).

Our proposed solution is two-phased: The first phase, to construct a structured document $\mathcal{X}^+$, augments the given document $\mathcal{X}$ by retrieving relevant keyphrases $\mathcal{R}$ from existing keyphrases in the training dataset. Then, the second phase follows, to encode structure-aware representations on $\mathcal{X}^+$. We use graph representation to effectively integrate $\mathcal{X}$ and $\mathcal{R}$, where the graph can be flexibly designed depending on closed/open set scenarios.

Our empirical results validate that generating and encoding document structures significantly improve performance, outperforming the state-of-the-art, for both social and scientific documents.

2 Related Work

We briefly explain our target task of KG and introduce our distinction of leveraging structures.

Observing words in the document is a crucial signal for keyphrase tasks, especially for keyphrase extraction (KE) requiring keyphrases to appear in the given document. In contrast, we focus on KG, as our target scenarios require not having such restrictions. KG, to generate absent words, is often modeled as neural encoder-decoder architecture (Sutskever et al., 2014), which generates the keyphrase sequence given the input document (Meng et al., 2017). KG approaches can be further categorized into two settings: one-to-one (O2O) and one-to-seq (O2S) (Yuan et al., 2020; Ye et al., 2021). In O2O, a model is trained to generate a single keyphrase for each document, and then, for evaluation, the model generates multiple keyphrases using beam search decoding with a large beam size (*e.g.*, 200). On the other hand, in O2S, a model is trained to generate multiple keyphrases where multiple keyphrases for each document are concatenated into a single sequence with a predefined delimiter. Our model follows O2O, as O2O is known to be better for predicting absent keyphrases (Meng et al., 2021) and as absent keyphrases are frequently observed in our target scenario with scarce context. The distinction of our proposal for KG is to leverage structures in documents, to improve both encoder and decoder.

Structures in documents are essential sources of prediction signals (Kim et al., 2021). For example, for a scientific publication consisting of a title and the main body, the structured document enjoys the complementary strength of the two fields: While the main body contains many keywords (*i.e.*, high recall), title, though much shorter, concisely describes the main focus of the paper (*i.e.*, high precision). To leverage such structure, TGNet (Chen et al., 2019b) uses the title to guide the encoder to accurately capture core contents. However, the given title is observed to be often insufficient due to length limitation (Li et al., 2010), which we consistently observed in our evaluations.

As a further extreme, short social media posts may not have titles. Furthermore, because of the length limitation (*e.g.*, 140 character tweets), most keywords may not appear in the given post (*i.e.*, low recall). In such scenarios, one can construct structured posts, to augment posts with missing keywords. For example, TAKG (Wang et al., 2019) utilizes topic modeling, where topics shared across other documents enable the encoder to leverage contexts observed in other related posts. However, a small, fixed number of topics (*e.g.*, 15 or 30) are limited in differentiating diverse documents with similar topics.

Our distinction is overcoming incomplete or missing structure, by generating a virtual structure from existing keyphrases (keyphrases in the train set), from which we "retrieve" terms that can serve as titles or topics for encoding. KG-KE-KR (Chen et al., 2019a) similarly leverages keyphrase retrieval, but they do not use this for structure-aware

encoding and use only for decoding. In contrast, we jointly contextualize the given document and the retrieved keyphrases, to allow both fields to exchange contexts from each other. We empirically validate that our proposed structure-augmented encoding significantly boosts performance.

3 Approach

Given a document $\mathcal{X} = \{x_i\}_{i=1}^{N_\mathcal{X}}$ with $N_\mathcal{X}$ unique words, KG models aim to output a set of target keyphrases, which can be implemented using encoder-decoder architecture. We train a model to predict a single keyphrase, then, for inference, we generate multiple keyphrases using beam search decoding. We denote a single keyphrase with length T by $\mathcal{Y} = [y_1, \ldots, y_T]$.

In the following sections, we first describe a baseline encoding targeting plain text $\mathcal{X}$ (§3.1), and then, we explain our distinction of generating structure (§3.2.1) and encoding structure (§3.2.2).

In this section, we do not consider pre-existing title $\mathcal{T}$, but our framework straightforwardly extends for such titles, as we later discuss in §4.4.1.

3.1 Baseline: Plain Text Encoding

Targeting plain texts, a novel graph-based encoder was proposed in DivGraphPointer (Sun et al., 2019), outperforming sequence-based encodings such as GRU (Cho et al., 2014) or LSTM (Hochreiter and Schmidhuber, 1997).

For graph construction, a fully connected graph is used, where nodes are unique words in $\mathcal{X}$. For edges, two adjacency matrices $\overleftarrow{\mathbf{A}}^{\mathcal{X}}_{ij}$ and $\overrightarrow{\mathbf{A}}^{\mathcal{X}}_{ij}$, using position-based proximity, are obtained for forward and backward direction respectively:

$$\begin{aligned}\overleftarrow{\mathbf{A}}^{\mathcal{X}}_{ij} &= \sum_{p_i^\mathcal{X}} \sum_{p_j^\mathcal{X}} max((p_i^\mathcal{X} - p_j^\mathcal{X})^{-1}, 0) \\ \overrightarrow{\mathbf{A}}^{\mathcal{X}}_{ij} &= \sum_{p_i^\mathcal{X}} \sum_{p_j^\mathcal{X}} max((p_j^\mathcal{X} - p_i^\mathcal{X})^{-1}, 0),\end{aligned} \tag{1}$$

where $p^\mathcal{X}_{[i/j]} \in \mathcal{P}(x_{[i/j]})$ is a position offset of a unique word $x_{[i/j]}$ in $\mathcal{X}$.

Given the graph, to contextualize node representations, Graph Convolutional Network (GCN) (Kipf and Welling, 2017) is adopted. We denote the number of stacked graph convolution layers by L, the number of node features by D, and contextualized representations of nodes for l-th layer by $\mathbf{H}^\mathcal{X}_l \in \mathbb{R}^{N_\mathcal{X} \times D}$ where $\mathbf{H}^\mathcal{X}_1$ is obtained from a word embedding matrix. As comprehensive notations, we denote learnable parameters by $\mathbf{v}$ for vector and $\mathbf{W}$ for matrix, with different super/subscripts.

Starting from $\mathbf{H}^\mathcal{X}_1$, context vectors $\mathbf{C}^\mathcal{X}_l \in \mathbb{R}^{N_\mathcal{X} \times D}$ are gathered from neighbor nodes using graph convolution, then combined with $\mathbf{H}^\mathcal{X}_l$ to produce $\mathbf{H}^\mathcal{X}_{l+1}$ (Dauphin et al., 2017):

$$\begin{aligned}\mathbf{C}^\mathcal{X}_l &= \overleftarrow{\hat{\mathbf{A}}}^\mathcal{X} \mathbf{H}^\mathcal{X}_l \overleftarrow{\mathbf{W}}^\mathcal{X}_l + \overrightarrow{\hat{\mathbf{A}}}^\mathcal{X} \mathbf{H}^\mathcal{X}_l \overrightarrow{\mathbf{W}}^\mathcal{X}_l \\ &\quad + \mathbf{H}^\mathcal{X}_l \tilde{\mathbf{W}}^\mathcal{X}_l \qquad (2) \\ \mathbf{H}^\mathcal{X}_{l+1} &= \mathbf{H}^\mathcal{X}_l + \mathbf{C}^\mathcal{X}_l \times \sigma(\mathbf{G}^\mathcal{X}_l), \qquad (3)\end{aligned}$$

where σ denotes sigmoid function, and $\overleftarrow{\hat{\mathbf{A}}}^\mathcal{X}$, $\overrightarrow{\hat{\mathbf{A}}}^\mathcal{X}$ are normalized matrices with eigenvalues close to 1 for stable training (Kipf and Welling, 2017). In Eq (3), residual addition is employed with $\mathbf{G}^\mathcal{X}_l \in \mathbb{R}^{N_\mathcal{X} \times D}$, which helps gradient back-propagation through deep layers (He et al., 2016; Dauphin et al., 2017). $\mathbf{G}^\mathcal{X}_l$ is obtained in the same way to $\mathbf{C}^\mathcal{X}_l$ in Eq (2) but with different learnable parameters.

We adopt the same graph construction and GCN contextualization, for encoding $\mathcal{X}$. Built upon the GCN encoder, our distinction is to generate and encode structure for $\mathcal{X}$.

3.2 Proposed: Structure-Augmented KG

Beyond plain text, we aim to leverage structures in documents. Though leveraging the title-body structure in scientific publications has been shown effective (Chen et al., 2019b), the given titles are often found to be incomplete because of limited length (Li et al., 2010), or unavailable (*e.g.*, tweets).

Our goal is, given incomplete or missing structure, to generate and encode structures, to replace missing titles or complement incomplete titles.

3.2.1 Generating Structure

To generate structured document $\mathcal{X}^+$, we leverage existing keyphrases of other documents, specifically similar documents to $\mathcal{X}$, by adopting an assumption that similar documents tend to have similar keyphrases. Specifically, for each keyphrase r in the training dataset, we first collect supporting documents, having r as one of the ground-truth keyphrases, then concatenate them as a single document, denoted by $\mathcal{S}_r$, and use $\mathcal{S}_r$ to index r. We then use BM25 search (Robertson and Walker, 1994) with $\mathcal{X}$ as a query, to retrieve top-K relevant keyphrases $\mathcal{R} = [r_1, \ldots, r_K]$[2]. Finally, we extend $\mathcal{X}$ with $\mathcal{R}$, to construct a structured document $\mathcal{X}^+$.

[2]We used pyserini (Lin et al., 2021) for retrieval.

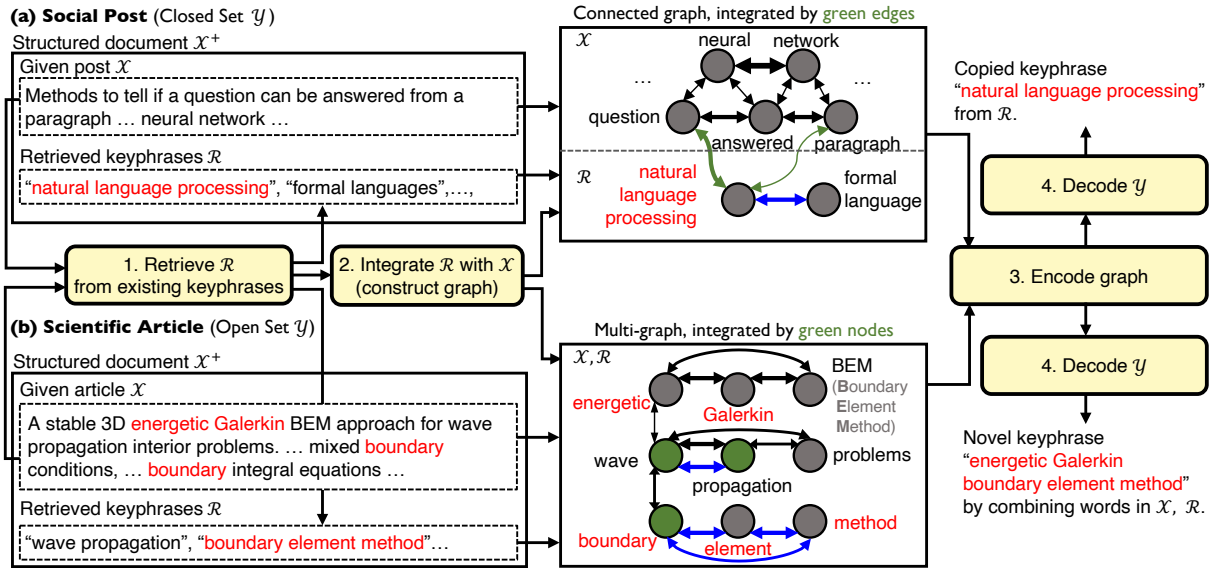

Figure 1: Overall approach: Red-colored words or phrases are texts included in decoded keyphrases. **Black** edges and blue edges are proximity between nodes, obtained from $\mathcal{X}$ and $\mathcal{R}$ respectively, where the thickness indicates the degree of relatedness between nodes. To construct an integrated graph for social posts, we connect two graphs (from $\mathcal{X}$ and $\mathcal{R}$), using green edges, and for scientific articles, we construct a multigraph with two types of edges (for $\mathcal{X}$ and $\mathcal{R}$), by merging nodes having the same words, depicted by green nodes.

Having relevant keyphrases $\mathcal{R}$ enhances both encoding and decoding phases, as described next.

3.2.2 Encoding Structure

Once we augment $\mathcal{X}$ with $\mathcal{R}$, our next step is to integrate $\mathcal{X}$ and $\mathcal{R}$ into $\mathcal{X}^+$, and jointly contextualize contents in $\mathcal{X}^+$ by exchanging contexts between the two fields.

For effective integration, we represent $\mathcal{X}^+$ as an **integrated graph**, that can be flexibly designed, based on the following principle: A pair of two highly related nodes should be connected by an edge or merged into a single node, while unrelated nodes should be separated from each other.

For encoding $\mathcal{X}$, we adopt the GCN contextualization, described in §3.1. On the other hand, for $\mathcal{R}$, graphs are differently encoded for open- and closed-set scenarios, as below.

- Closed set $\mathcal{Y}$: Keyphrases in $\mathcal{R}$ are likely to be reused for the target keyphrases $\mathcal{Y}$, where *keyphrase* in $\mathcal{R}$ is assigned as a node. The keyphrase can be copied as $\mathcal{Y}$ based on the node representation.
- Open set $\mathcal{Y}$: Keyphrases may not be reused, such that words in $\mathcal{R}$ should be combined with other words to generate novel keyphrases, for which we assign the node with a *keyword* in $\mathcal{R}$.

In the following two sections, we introduce our **G**raph-based **S**tructured Document **Enc**oder, or **GSEnc**, specifically for closed (§3.2.3) and open set $\mathcal{Y}$ (§3.2.4) respectively.

3.2.3 GSEnc for closed set $\mathcal{Y}$

Targeting closed set keyphrases, we jointly contextualize word nodes from $\mathcal{X}$ and keyphrase nodes from $\mathcal{R}$, by building a graph each, then generate connecting edges between the two for propagating contexts across graphs.

Constructing and encoding graph for $\mathcal{X}$ follow §3.1. On the other hand, for $\mathcal{R}$, we set a single phrase node for each keyphrase, instead of multiple word nodes, to enable a decoder to copy the keyphrase as-is based on the node features. For edge construction, instead of position-based proximity in Eq (1) which is not available for keyphrase nodes, we adopt co-occurrence-based proximity, as frequently co-occurred keyphrases tend to have similar topics[3]. Specifically, we compute adjacency matrix $\mathbf{A}^{\mathcal{R}}$ as conditional probabilities based on co-occurrence between keyphrases, then, contexts are gathered using graph convolution, similar to Eq (2).

$$\mathbf{A}^{\mathcal{R}}_{jk} = p(r_k|r_j) \tag{4}$$

$$\mathbf{C}^{\mathcal{R}}_{l} = \hat{\mathbf{A}}^{\mathcal{R}}\mathbf{H}^{\mathcal{R}}_{l}\hat{\mathbf{W}}^{\mathcal{R}}_{l} + \mathbf{H}^{\mathcal{R}}_{l}\tilde{\mathbf{W}}^{\mathcal{R}}_{l}, \tag{5}$$

where $\mathbf{H}^{\mathcal{R}}_{l} \in \mathbb{R}^{K\times D}$ is node features for $\mathcal{R}$ in l-th layer, $p(r_k|r_j)$ is the probability that r_k co-

[3]We say two keyphrases co-occur when the two keyphrases are gold keyphrases for the same document.

occurs given r_j, which is computed using the training dataset, and $\hat{\mathbf{A}}^{\mathcal{R}}$ is normalized matrix from $\mathbf{A}^{\mathcal{R}}$, as in $\hat{\mathbf{A}}^{\mathcal{X}}$ from $\mathbf{A}^{\mathcal{X}}$.

We now discuss how to connect the two graphs for $\mathcal{X}$ and $\mathcal{R}$ (green edges in Figure1). For such connection, we are inspired by connecting query (keyphrase in our case) and document, using relevance feedbacks, such as clicks on matching query-document pair. As we have no such feedback between each keyphrase $r_j \in \mathcal{R}$ and $\mathcal{X}$, we can adopt zero-shot query log synthesis (Ma et al., 2021), by using *pseudo*-relevance feedback (Rocchio, 1971): We treat supporting documents $\mathcal{S}_{r_j}$ as feedback documents with pseudo-relevance to r_j, to connect r_j with $\mathcal{X}$ using top-M overlapped words, denoted by $\mathcal{S}^*_{r_j}$, between $\mathcal{S}_{r_j}$ and $\mathcal{X}$. For selecting top-M, an unsupervised signal frequently used is tf-idf, to favor words appearing frequently in $\mathcal{S}_{r_j}$ (*i.e.*, representative words) but infrequently in other documents (*i.e.*, discriminative words) (Xu and Croft, 2017). Alternatively, query generators can be supervised as a separate task, requiring additional training data, so we adopt the former.

Given $\mathcal{S}^*_{r_j}$, we construct an edge between x_i and r_j, if and only if x_i is included in $\mathcal{S}^*_{r_j}$, then we enable the encoder to estimate a proper edge weight $\overleftrightarrow{\mathbf{A}}_{ij}$, using graph attention network (Veličković et al., 2018)[4], such that only relevant contexts are exchanged between $\mathcal{X}$ and $\mathcal{R}$.

$$\overleftrightarrow{\mathbf{A}}_{ij} = f(\overleftrightarrow{\mathbf{v}}^\top[\overleftrightarrow{\mathbf{W}}\mathbf{H}^{\mathcal{X}}_{i,l}||\overleftrightarrow{\mathbf{W}}\mathbf{H}^{\mathcal{R}}_{j,l}]) \quad (6)$$

$$\mathbf{C}^{\mathcal{X}\to\mathcal{R}}_l = \text{softmax}(\overleftrightarrow{\mathbf{A}}^\top)\overleftrightarrow{\mathbf{W}}\mathbf{H}^{\mathcal{X}}_l \in \mathbb{R}^{K\times D} \quad (7)$$

$$\mathbf{C}^{\mathcal{R}\to\mathcal{X}}_l = \text{softmax}(\overleftrightarrow{\mathbf{A}})\overleftrightarrow{\mathbf{W}}\mathbf{H}^{\mathcal{R}}_l \in \mathbb{R}^{N_{\mathcal{X}}\times D}, \quad (8)$$

where $f(\cdot)$ is LeakyReLU nonlinearity (Maas et al., 2013), and "||" denotes concatenation. In addition, we further leverage pseudo-relevance feedbacks, to augment contexts of r_j, by using $\mathcal{S}_{r_j}$ to initialize node features for r_j: $\mathbf{H}^{\mathcal{R}}_{1,j} = \mathbf{E}^\top f(\mathcal{S}_{r_j}) \in \mathbb{R}^D$, where $\mathbf{E}$ is a trainable word embedding matrix, and f weighs each word in $\mathcal{S}_{r_j}$. For f, we adopt tf-idf[5].

Finally, we add all gathered contexts and combine these with $\mathbf{H}^{[\mathcal{X}/\mathcal{R}]}_l$, similar to Eq (3).

$$\begin{aligned}\mathbf{H}^{\mathcal{X}}_{l+1} &= \mathbf{H}^{\mathcal{X}}_l + \mathbf{C}^{\mathcal{X}}_l \times \sigma(\mathbf{G}^{\mathcal{X}}_l) \\ &\quad + \mathbf{C}^{\mathcal{R}\to\mathcal{X}}_l \times \sigma(\mathbf{G}^{\mathcal{R}\to\mathcal{X}}_l)\end{aligned} \quad (9)$$

$$\begin{aligned}\mathbf{H}^{\mathcal{R}}_{l+1} &= \mathbf{H}^{\mathcal{R}}_l + \mathbf{C}^{\mathcal{R}}_l \times \sigma(\mathbf{G}^{\mathcal{R}}_l) \\ &\quad + \mathbf{C}^{\mathcal{X}\to\mathcal{R}}_l \times \sigma(\mathbf{G}^{\mathcal{X}\to\mathcal{R}}_l),\end{aligned} \quad (10)$$

where the $\mathbf{G}$s are obtained in the same way to $\mathbf{C}$s but with different learnable parameters.

3.2.4 GSEnc for open set $\mathcal{Y}$

Targeting open set keyphrases, we jointly contextualize word nodes from both $\mathcal{X}$ and $\mathcal{R}$.

Since nodes for $\mathcal{X}$ and $\mathcal{R}$ have the same granularity (*i.e.*, word-level nodes), instead of connecting two graphs (for $\mathcal{X}$ and $\mathcal{R}$) using edges, we construct a single integrated graph, by merging a node for $\mathcal{X}$ and a node for $\mathcal{R}$ whenever the two nodes correspond to the same word (green nodes in Figure1). Such integration enables contexts scattered across the two fields to be effectively gathered into the merged node, from which other nodes connected to the merged node also can exchange their contexts with each other through the merged node. Though sharing nodes, we use separate edges for $\mathcal{X}$ and $\mathcal{R}$, for structure-aware encoding. That is, the merged graph becomes a multigraph; we connect a pair of nodes with two types of edges (black edges for $\mathcal{X}$ and blue edges for $\mathcal{R}$ in Figure1). We denote the contextualized features for nodes on the merged graph, including nodes from both $\mathcal{X}$ and $\mathcal{R}$, by $\mathbf{H}_l^{\mathcal{X}^+_{\text{merged}}}$.

As in Eq (2), using graph convolution and the two types of edges, we aggregate neighbor contexts from $\mathcal{X}$ and $\mathcal{R}$, into $\mathbf{C}_l^{\mathcal{X}_{\text{merged}}}$ and $\mathbf{C}_l^{\mathcal{R}_{\text{merged}}}$ respectively, while obtaining $\mathbf{G}_l^{\mathcal{X}_{\text{merged}}}$,$\mathbf{G}_l^{\mathcal{R}_{\text{merged}}}$ similarly. For adjacency matrix for keywords in $\mathcal{R}$, we use position-based proximity between two keywords within each keyphrase in $\mathcal{R}$, as in Eq (1). Then, we combine both contexts to update $\mathbf{H}_l^{\mathcal{X}^+_{\text{merged}}}$.

$$\begin{aligned}\mathbf{H}^{\mathcal{X}^+_{\text{merged}}}_{l+1} &= \mathbf{H}^{\mathcal{X}^+_{\text{merged}}}_l \\ &\quad + \mathbf{C}^{\mathcal{X}_{\text{merged}}}_l \times \sigma(\mathbf{G}^{\mathcal{X}_{\text{merged}}}_l) \\ &\quad + \mathbf{C}^{\mathcal{R}_{\text{merged}}}_l \times \sigma(\mathbf{G}^{\mathcal{R}_{\text{merged}}}_l).\end{aligned} \quad (11)$$

Given the contextualized node features at the last GCN layer, such as $(\mathbf{H}^{\mathcal{X}}_L, \mathbf{H}^{\mathcal{R}}_L)$ or $\mathbf{H}_L^{\mathcal{X}^+_{\text{merged}}}$, for closed set $\mathcal{Y}$ or open set $\mathcal{Y}$ respectively, we feed the features into the decoder, to generate keyphrases.

[4]We use single-head attention. In our experiments, performances between single/multi-head attention were comparable.

[5]Since the number of documents in $\mathcal{S}_{r_j}$ can vary among keyphrases, we normalized tf-idf weights to have unit norm.

3.3 Decoder

The goal of the decoder is to generate a target keyphrase $\mathcal{Y}=[y_1,\ldots,y_T]$ with length T, based on the contextualized features on $\mathcal{X}^+$ from the encoder. While standard KG models either copy words from $\mathcal{X}$ or generate words from the predefined vocabulary $\mathcal{V}$, we have $\mathcal{R}$ as an additional source, which provides valid keyphrase candidates already used for similar documents. To leverage $\mathcal{R}$ for decoding, depending on the application, we allow the decoder to copy either (a) keyphrases (closed set $\mathcal{Y}$) or (b) words (open set $\mathcal{Y}$), from $\mathcal{R}$.

(a) Closed set $\mathcal{Y}$: copying keyphrases In social media posts, a few keyphrases (*e.g.*, trending hashtags) cover most of the potential target keyphrases. Thus, when $\mathcal{Y}$ is included in $\mathcal{R}$, copying a keyphrase (*e.g.*, "natural language processing" in Figure 1) from $\mathcal{R}$ makes efficient decoding.

To copy relevant keyphrases, we compute relevance score ϕ_j of each $r_j\in\mathcal{R}$ to $\mathcal{X}$, using inner product between $\mathbf{H}^{\mathcal{R}}_{L,j}$ and the summarized features for $\mathcal{X}$, denoted by $\tilde{\mathbf{h}}^{\mathcal{X}}$:

$$\phi_j = \mathbf{H}^{\mathcal{R}}_{L,j}{}^{\top}\tilde{\mathbf{h}}^{\mathcal{X}}, \tilde{\mathbf{h}}^{\mathcal{X}} = \sum_{i=1}^{N_{\mathcal{X}}} \alpha_i \mathbf{H}^{\mathcal{X}}_{L,i} \quad (12)$$

$$\alpha_i = \text{softmax}_i(\mathbf{v}^{\top}_{\text{pool}}\tanh(\mathbf{W}_{\text{pool}}\mathbf{H}^{\mathcal{X}}_{L,i})). \quad (13)$$

Given $\{\phi_j\}_{j=1}^K$, we copy top-ranked keyphrases among $\mathcal{R}$, regarding $\{\phi_j\}_{j=1}^K$. For training, we use mean square error (MSE)[6] as the objective function: $\mathcal{L}_{\text{MSE}} = \sum_{j=1}^K ||\phi_j - \phi^*_j||^2_2$, where $\phi^*_j = 1$ if $r_j = \mathcal{Y}$ and $\phi^*_j = 0$ otherwise.

(b) Open set $\mathcal{Y}$: copying keywords On the other hand, in scientific publications, copying a word from $\mathcal{R}$ (*e.g.*, "element", "method" in Figure 1) enables the decoder to generate novel keyphrases (*e.g.*, "energetic Galerkin boundary element method"), by combining the word with words from the other two sources (*e.g.*, "energetic Galerkin" from $\mathcal{X}$).

We adopt a single layer GRU decoder equipped with copy mechanism (See et al., 2017). For simplicity, we denote $\mathbf{H}^{\mathcal{X}^+_{\text{merged}}}_L$ by $\mathbf{H}$ and the number of nodes in the merged graph by N. For t-th word (*i.e.*, y_t) decoding, copy scores p^{copy}_t are computed using attention: $p^{\text{copy}}_{t,k} = \text{softmax}(\mathbf{v}^{\top}_1\mathbf{W}_1[\mathbf{H}_k||\mathbf{o}_t])$, where $\mathbf{o}_t$ denotes the decoder hidden state. However, p^{copy}_t only covers y_t present in $\mathcal{X}^+$. To predict arbitrary y_t using $\mathcal{V}$, generation scores p^{gen}_t are computed by $p^{\text{gen}}_t = \text{softmax}(\mathbf{E}_{\mathcal{V}}(\mathbf{W}_2[\hat{\mathbf{h}}_t||\mathbf{o}_t]))$, where $\mathbf{E}_{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}|\times D}$ is a learnable word embedding matrix for $\mathcal{V}$, and $\hat{\mathbf{h}}_t$ ($=\sum_{k=1}^N p^{\text{copy}}_{t,k}\mathbf{H}_k$) summarizes relevant contexts from $\mathcal{X}^+$, using p^{copy}_t as relevance scores. The final score p^{final}_t is computed as a combination of the two scores using a gate value z_t: $p^{\text{final}}_t = p^{\text{copy}}_t \times z_t + p^{\text{gen}}_t \times (1-z_t)$, and $z_t = \sigma(\mathbf{v}^{\top}_3[\hat{\mathbf{h}}_t||\mathbf{o}_t])$. Once y_t is decoded according to p^{final}_t, we update the decoder hidden state $\mathbf{o}_t$: $\mathbf{o}_{t+1} = \text{GRU}(\mathbf{y}_t, \mathbf{o}_t)$, where $\mathbf{y}_t\in\mathbb{R}^D$ is the feature vector for the decoded token y_t.

For $\mathbf{y}_t$, previous work uses a word embedding matrix. However, when y_t is a rare word, which is often replaced by "[UNK]" symbol, $\mathbf{y}_t$ from the embedding matrix contains little information on the word. We thus leverage structure-aware representation $\mathbf{H}$, to capture the meaning of y_t based on the contexts within $\mathcal{X}^+$. Specifically, when y_t is copied from $\mathcal{X}^+$, similar to p^{final}_t, we compute $\mathbf{y}_t$ as combination of $\mathbf{H}$ and $\mathbf{E}_{\mathcal{V}}$ from $\mathcal{X}^+$ and $\mathcal{V}$ respectively: $\mathbf{y}_t = \mathbf{H}_{y_t\in\mathcal{X}^+}\times z_t + \mathbf{E}_{y_t\in\mathcal{V}}\times(1-z_t)$, where $\mathbf{H}_{y_t\in\mathcal{X}^+}$, $\mathbf{E}_{y_t\in\mathcal{V}}$ are corresponding vectors for y_t from $\mathbf{H}$ and $\mathbf{E}_{\mathcal{V}}$ respectively.

Following convention (Meng et al., 2017), to predict multiple keyphrases, we use beam search with beam size 200, and then use top-ranked keyphrases as final predictions. For training, we use cross entropy loss on y_t as the objective function.

4 Experiment

4.1 Dataset

Dataset	# of D	# of unique KP	# of KP per D	% of abs KP
Twitter	44K	4K	1.13	71.4
Weibo	46K	2K	1.06	75.7
SE	49K	12K	2.43	54.3
KP20k	510K	670K	2.94	36.7

Table 2: Statistics for three social datasets and a scientific publication dataset. "**D**" and "**KP**" are short for document and keyphrases respectively. "**abs KP**" denotes absent keyphrases. SE denotes StackExchange.

We conducted experiments on social media posts and scientific publication, with missing or incomplete structures. We present statistics on the datasets in Table 2.

For social media posts, we used three public datasets including not only microblog posts such

[6] We also tried cross entropy, but found the MSE to be better empirically.

as **Twitter** and **Weibo**, but also a Q&A platform such as **StackExchange** (Wang et al., 2019)[7]. In the Twitter and Weibo datasets, user-assigned hashtags are treated as keyphrases of a corresponding post. Hashtags in the middle of a post were treated as present keyphrases, and hashtags either before or after a post were treated as absent keyphrases and are removed from the post. For the StackExchange dataset, a given document is a question, and keyphrases are manually annotated by users. Different from microblogs, for each question, there is a title and a description such that the given document is a concatenation of the title and the description for the question. For training and evaluation, document-keyphrase pairs are partitioned into train, validation, and test splits consisting of 80%, 10%, and 10% of the entire data respectively.

As in (Wang et al., 2019), we adopted macro-averaged F1@k and mean average precision (mAP) as evaluation metrics. To compute F1@k, we count the number of correct keyphrases among the top-k predictions (denoted by $hit@k$), then, precision and recall are computed as $hit@k$ divided by the number of predictions (*i.e.*, k) and the number of gold keyphrases respectively. We report F1@1/3 for Twitter/Weibo and F1@3/5 for StackExchange considering the average number of keyphrases in datasets (1.13, 1.06, and 2.43 respectively). For all datasets, we report mAP over the top-5 predictions.

For scientific publication, we use **KP20k** dataset (Meng et al., 2017). Since the original dataset includes duplicates between train/test documents, we use a preprocessed dataset without duplicates, released by (Chen et al., 2019a)[8]. The dataset has 510K, 20K, 20K documents for train, validation, and test datasets respectively. As in the baselines, we use F1@5/10 as evaluation metrics.

For all datasets, following (Meng et al., 2017), we applied stemming using Porter Stemmer (Porter, 1980)[9] as preprocessing, where duplicate keyphrases were removed after stemming.

4.2 Baselines

We compared our models with previous state-of-the-art KG baselines as well as KE baselines.

For KE baselines, we use TextRank (Mihalcea and Tarau, 2004) and TF-IDF which are the most popular unsupervised keyword ranking models, and a neural sequence tagging model (Zhang et al., 2016) (denoted by Seq-Tag) that predicts keyphrase spans within the given document.

For KG baselines, we use CopyRNN (Meng et al., 2017) and CorrRNN (Chen et al., 2018) in common for both social media posts and scientific publication. CopyRNN adopts standard encoder-decoder architecture with copy mechanism. CorrRNN exploits correlation between keyphrases to predict diverse keyphrases. We compare our proposed model with the previous state-of-the-art baselines, including TAKG (Wang et al., 2019) for social media post datasets, that augments contexts of the given posts using topic modeling, and KG-KE-KR-M (Chen et al., 2019a) for KP20k, that uses the retrieved keyphrases $\mathcal{R}$ to provide the decoder with additional contexts. Note that, different from ours, KG-KE-KR-M separately encodes $\mathcal{X}$ and $\mathcal{R}$, thus do not enjoy structure-aware representations.

To validate the effectiveness of joint contextualization of $\mathcal{X}$ and $\mathcal{R}$ on the graph, we also conduct an ablation study, where we compare our proposed model to the ablation model that separately encodes the graph for $\mathcal{X}$ and the graph for $\mathcal{R}$ (denoted by w/o integration). For social media posts, we exclude all edges between $\mathcal{X}$ and $\mathcal{R}$ (green edges in Figure 1). For scientific publication, we separately encode two graphs (each for $\mathcal{X}$ or $\mathcal{R}$) instead of a merged multigraph. The ablation model is similar to KG-KE-KR-M that also separately encodes $\mathcal{X}$ and $\mathcal{R}$, while differing in the encoder (graph encoder, instead of sequence encoder).

4.3 Implementation Details

For fair comparisons, we use the same hyperparameters and training strategy to the baselines, such as the size of the predefined vocabulary $\mathcal{V}$, batch size of 64, Adam optimizer (Kingma and Ba, 2015) with 0.001 initial learning rate, and gradient clipping (Pascanu et al., 2013) with 1.0 threshold.

As exceptions, we set the best value for D, L, K, and M, based on validation performance. The best D, among $\{150, 300\}$, was 300 for all datasets. The best L, among $\{2, 3, 4, 5\}$, was 2 for social datasets and 3 for KP20k. The best K, among $\{10, 20, 30, 40, 50\}$, was 20/20/30, for Twitter/Weibo/StackExchange respectively. Because of the larger number of keyphrases per document in StackExchange (2.43) than Twitter (1.13) and Weibo (1.06), K should be larger in StackEx-

[7] https://github.com/yuewang-cuhk/TAKG

[8] https://github.com/Chen-Wang-CUHK/KG-KE-KR-M

[9] https://www.nltk.org/_modules/nltk/stem/porter.html

Model	Social media post									Scientific articles	
	Twitter			Weibo			StackExchange			KP20k	
	F1@1	F1@3	mAP	F1@1	F1@3	mAP	F1@3	F1@5	mAP	F1@5	F1@10
KE baselines											
TF-IDF	1.2	1.1	1.9	1.9	1.5	2.5	13.5	12.7	12.6	8.7	11.3
TextRank	1.7	1.9	1.9	0.2	0.5	0.6	6.0	8.3	4.8	15.1	13.2
Seq-Tag	22.8_{3}	12.3_{2}	22.4_{3}	16.3_{2}	9.0_{1}	16.5_{3}	17.6_{16}	12.8_{12}	19.0_{13}	-	-
KG w/o structure											
CopyRNN	36.6_{11}	26.8_{5}	43.1_{12}	32.0_{3}	22.7_{2}	38.0_{1}	31.5_{3}	27.4_{2}	33.5_{1}	30.6_{1}	27.3_{0}
CorrRNN	35.0_{8}	26.1_{4}	41.6_{5}	31.6_{7}	22.2_{5}	37.5_{8}	30.9_{3}	27.0_{2}	32.9_{6}	29.1_{2}	26.4_{2}
KG-KE-KR-M	-	-	-	-	-	-	-	-	-	31.7_{0}	28.2_{0}
KG w/ structure											
TAKG (topic)	38.5_{3}	27.8_{0}	45.1_{2}	35.0_{3}	24.4_{2}	41.3_{4}	33.4_{2}	29.2_{1}	35.5_{1}	-	-
TGNet (title)	-	-	-	-	-	-	32.0_{3}	27.8_{3}	34.1_{4}	31.4_{3}	28.1_{3}
KG w/ structure (ours)											
GSEnc (keyphrase)	38.8_{7}	28.1_{2}	45.5_{6}	$\mathbf{42.8}_{6}$	$\mathbf{29.5}_{3}$	$\mathbf{50.0}_{4}$	$\mathbf{35.2}_{3}$	$\mathbf{30.9}_{2}$	$\mathbf{37.8}_{3}$	$\mathbf{32.9}_{3}$	$\mathbf{29.0}_{2}$
w/o integration	36.8_{4}	27.0_{5}	43.3_{2}	38.6_{4}	27.0_{3}	45.7_{3}	33.2_{3}	29.3_{2}	35.5_{2}	31.9_{3}	27.9_{2}

Table 3: Keyphrase prediction performance. **Bold-face** indicates the best performance with significance ($p < 0.05$ using Student's paired t-test). We report average performance with standard deviation with different random seeds of 5 runs (*e.g.*, 29.4_{1} indicates 29.4 ± 0.1).

change. For fair comparison with KG-KE-KR-M that uses the retrieved keyphrases, we used the same $\mathcal{R}$ used in KG-KE-KR-M, for KP20k. For M for social media posts (§3.2.3), we search the best value among $\{10, 25, 50, 100\}$. The best M was 50 for both Twitter and Weibo, while it was 25 for StackExchange. Since documents are longer in StackExchange (88 words on average) than Twitter (20) and Weibo (33) with length limitation, smaller M makes more conservative edge construction.

Our experiments were conducted using a single GeForce RTX 2080Ti NVIDIA GPU. We used Pytorch (Paszke et al., 2019) for implementation.

4.4 Evaluation

In this section, we confirm the effectiveness of structures in documents for KG, and validate the superiority of the proposed structure over other structures. Results are shown in Table 3.

Since large portion of ground-truth keyphrases are absent keyphrases (*i.e.*, context-scarce $\mathcal{X}$), KE models show significantly worse performances compared to KG models, on both datasets.

Among KG models, structures in document significantly improve performances, for both datasets, where TAKG with latent topics and TGNet with title significantly outperform the other KG models without structures, except KG-KE-KR-M on KP20k. However, both a small number of latent topics and short titles have limited information. In contrast, by retrieving relevant keyphrases from existing keyphrases, we augment the given document with sufficient topical information. As a result, GSEnc outperforms all baselines, except TAKG on Twitter with comparable performance.

For social datasets, in addition to having superior or comparable performance over baselines, our proposed model, by avoiding sequential decoding, requires much lower computational costs than other KG models. Regarding computational efficiency, our model can process each text in about 22 ms (21 ms for retrieving $\mathcal{R}$ and less than 1 ms for the rest) while other KG approaches consume about 90 ms.

Meanwhile, we stress that it is important to jointly contextualize $\mathcal{X}$ and $\mathcal{R}$, to enjoy mutual benefits between them; we found that performance significantly decrease on all datasets, when we separately encode them (w/o integration in Table 3).

4.4.1 Integrating $\mathcal{R}$ with given titles

Model	Input	KP20k	SE
		F1@3	F1@5
TGNet	$\mathcal{X}$+$\mathcal{T}$	31.4	32.0
GSEnc (ours)	$\mathcal{X}$+$\mathcal{R}$	**32.9**	35.4
	$\mathcal{X}$+$\mathcal{R}$+$\mathcal{T}$	**32.9**	**36.3**

Table 4: Evaluation results using different structures: SE denotes for StackExchange dataset.

When titles are available, we can use $\mathcal{R}$, to complement concise, but incomplete titles (high precision, low recall), by providing missing terms in the title (increasing recall). In this section, we explain how to integrate $\mathcal{R}$ and title (denoted by $\mathcal{T}$), for better structured documents, and evaluate the effectiveness of such integration. We use KP20k and StackExchange datasets where titles of documents are written by the authors of the documents.

Since the title words are already represented as nodes in $\mathcal{X}$, we can simply add another edge type,

as similarly done in §3.2.4, to make three types of edges for the same node pair: edges from $\mathcal{X}$, $\mathcal{R}$, and the given title respectively. As in edges for $\mathcal{X}$, we use position-based proximity between title words, for edge weights. Given the title edges, we use contexts gathered from the title, to update contextualized representations, similar to Eq (3,9). The results are shown in Table 4.

We observe that $\mathcal{R}$ is more representative in KP20k than StackExchange, where the F1 accuracy of $\mathcal{R}$ on gold keywords was 22.3 and 11.5, for KP20k and StackExchange respectively. This can be explained by the document length difference: When $\mathcal{R}$ is less relevant, for shorter documents in StackExchange, titles with high precision capture relevant keyphrases in $\mathcal{R}$, such that $\mathcal{X}$+$\mathcal{R}$+$\mathcal{T}$ outperforms $\mathcal{X}$+$\mathcal{R}$, while $\mathcal{X}$+$\mathcal{R}$ is sufficiently accurate, otherwise. Our proposed approach, by leveraging both fields, work well in both cases, outperforming TGNet that uses only title structure.

4.4.2 Future work

Model (**Input**)	ID test	OOD test
	KP20k	Inspec
CopyRNN ($\mathcal{X}$)	30.6	**25.1**
GSEnc ($\mathcal{X}$+$\mathcal{R}$)	**32.9**	24.4

Table 5: Different effects of leveraging $\mathcal{R}$, for in-distribution (ID) and out-of-distribution (OOD) test datasets. We present F1@5 performance of CopyRNN and GSEnc (ours), that uses $\mathcal{X}$ and $\mathcal{X} + \mathcal{R}$ as inputs, respectively.

Leveraging existing keyphrases in train set is specifically effective when test distribution is similar to the train distribution, *i.e.*, in-distribution test set, as we validated in our experiments. On the other hand, for out-of-distribution test set, where train/test distributions are different from each other, existing keyphrases may not be helpful. We empirically tested different effects of leveraging $\mathcal{R}$, by training models using KP20k training dataset and evaluating the models on KP20k test set (in-distribution) and Inspec (Hulth, 2003) test set (out-of-distribution). While, in KP20k datasets, keyphrases are labeled by authors of documents, keyphrases in Inspec are labeled by third-party annotators. In Table 5, we can observe that $\mathcal{R}$ does not improve on out-of-distribution test set, while showing significant improvements on in-distribution test set.

To overcome, as a future work, our retriever can be extended to use external sources such as open-domain knowledge graphs (Shi et al., 2017) or Wikipedia texts (Yu and Ng, 2018), that generalize well to out-of-distribution data. However, such external sources will be less effective than internal sources (*e.g.*, existing keyphrases), when most of target documents are in-distribution documents. We will explore effective integration strategies between internal/external sources, to enjoy complementary strengths of those, for better performance on both in-distribution and out-of-distribution documents.

5 Conclusion

We studied the problem of KG for scientific text and social posts, representing context-scarce scenarios with open and closed keyphrase set, respectively. Our work is two-phased, for augmenting and encoding missing/incomplete structure. Empirical evaluation results validate that our proposed model outperforms the state-of-the-art in both problems.

Acknowledgments

This work is supported by IITP grants (ITRC, 2021-2020-0-01789, 2021-2016-0-00464) and SNU AI Graduate School Program (2021-0-01343).

References

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, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. 2019a. An integrated approach for keyphrase generation via exploring the power of retrieval and extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2846–2856, Minneapolis, Minnesota. Association for Computational Linguistics.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019b. Title-guided encoding for keyphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6268–6275.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of*

the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

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

Jihyuk Kim, Young-In Song, and Seung-won Hwang. 2021. Web document encoding for structure-aware keyphrase extraction. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1823–1827.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Decong Li, Sujian Li, Wenjie Li, Wei Wang, and Weiguang Qu. 2010. A semi-supervised key phrase extraction approach: learning from title phrases through a document semantic network. In *Proceedings of the ACL 2010 conference short papers*, pages 296–300.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: An easy-to-use python toolkit to support replicable ir research with sparse and dense representations. *arXiv preprint arXiv:2102.10073*.

Ji Ma, Ivan Korotkov, Yinfei Yang, Keith Hall, and Ryan McDonald. 2021. Zero-shot neural passage retrieval via domain-targeted synthetic question generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1075–1088.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer.

Rui Meng, Xingdi Yuan, Tong Wang, Sanqiang Zhao, Adam Trischler, and Daqing He. 2021. An empirical study on neural keyphrase generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4985–5007, Online. Association for Computational Linguistics.

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.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318. JMLR.org.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*, pages 232–241. Springer.

Joseph Rocchio. 1971. Relevance feedback in information retrieval. *The Smart retrieval system-experiments in automatic document processing*, pages 313–323.

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.

Wei Shi, Weiguo Zheng, Jeffrey Xu Yu, Hong Cheng, and Lei Zou. 2017. Keyphrase extraction using knowledge graphs. *Data Science and Engineering*, 2(4):275–288.

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.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Yue Wang, Jing Li, Hou Pong Chan, Irwin King, Michael R. Lyu, and Shuming Shi. 2019. Topic-aware neural keyphrase generation for social media language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2516–2526, Florence, Italy. Association for Computational Linguistics.

Jinxi Xu and W Bruce Croft. 2017. Quary expansion using local and global document analysis. In *Acm sigir forum*, volume 51, pages 168–175. ACM New York, NY, USA.

Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021. One2Set: Generating diverse keyphrases as a set. 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 4598–4608, Online. Association for Computational Linguistics.

Yang Yu and Vincent Ng. 2018. Wikirank: Improving keyphrase extraction based on background knowledge. *arXiv preprint arXiv:1803.09000*.

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.

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