Topic and Reference Guided Keyphrase Generation from Social Media

Chapter · July 2022

DOI: 10.1007/978-3-031-10986-7_12

5 authors, including:

Xiubin Yu
National University of Defense Technology
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Zhen Huang
National University of Defense Technology
90 PUBLICATIONS **875** CITATIONS
SEE PROFILE

Topic and Reference Guided Keyphrase Generation from Social Media

Xiubin Yu[1], Xingjun Chen[2] , Zhen Huang[1(✉)], Yong Dou[1], and Biao Hu[1]

[1] National Key Laboratory of Parallel and Distributed Processing, National University of Defense Technology, Changsha, China
{yuxiubin20,huangzhen,yongdou,hubiao}@nudt.edu.cn
[2] Dalian Navy Academy, Dalian, China

Abstract. Automatic keyphrase generation can help human efficiently understand or process critical information from massive social media posts. Seq2Seq-based generation models that can produce both present and absent keyphrases have achieved remarkable performance. However, existing models are limited by the sparseness of posts that are widely exhibited in social media language. Sparseness makes it difficult for models to obtain useful features and generate accurate keyphrases. To address this problem, we propose the Topic and Reference Guided Keyphrase Generation model(TRGKG) for social media posts, which enrich scarce posts features by corpus-level topics and post-level reference knowledge. The proposed model incorporates a contextual neural topic model to exploit topics and a heterogeneous graph to capture reference knowledge from retrieved related posts. To guide the decoding process, we introduce new topic-aware hierarchical attention and copy mechanism, which directly copies appropriate words from both the source post and its references. Experiments on two public datasets demonstrate that TRGKG achieves state-of-the-art performance.

Keywords: Keyphrase generation · Seq2Seq · Neural topic model · Heterogeneous graph

1 Introduction

Social media has become the main way for people to spread and obtain information. Hundreds of millions of social media posts are generated every day. To acquire knowledge from massive social media data, an automated system that can figure out important information is urgently needed. Keyphrase as the smallest unit of semantic expression can accurately describe the central concept or topic of a post. Social media keyphrases are beneficial to a wide range of applications, such as analyzing social behavior [13], constructing knowledge graph for social network [4], summarizing public opinions [9], etc. We denote phrases that do not match any contiguous subsequence of source text as absent keyphrases, and the ones that fully match a part of the text as present keyphrases.

Keyphrase Generation (KG) is a classic and challenging task that aims at predicting a set of keyphrases including present keyphrases as well as

G. Memmi et al. (Eds.): KSEM 2022, LNAI 13369, pp. 140–154, 2022.
https://doi.org/10.1007/978-3-031-10986-7_12

absent keyphrases for the given text. Exiting generation models will inevitably encounter the data sparsity issue when adapted to social media. It is essentially due to the informal and colloquial nature of social media language, which results in limited features available in the noisy data. To address this, TAKG [17] utilizes topic modeling, where corpus-level latent topics shared with across other documents enable the model to leverage contexts observed in other related posts. Nonetheless, TAKG regards the topic model as a separate component for topic extraction rather than jointly improving KG task and topic modeling in a unified way. Besides, the input of the neural topic model is a BoW (Bag-of-Words) vector, which does not contain the sequence information of the post. [12] shows that the features encoded using a sequence model can help generate more coherent topics. GSEnc [5] argument the title of the post by retrieving absent keywords to enrich the scarce post. These retrieved keywords are regarded as post-level reference knowledge. Reference knowledge can enrich spare training data so that the model can accurately memorize and summarize all candidates. However, the reference knowledge in GSEnc is only from retrieved keywords that lack semantic information, and the potential knowledge in the retrieved posts is ignored.

Thus, we propose a new keyphrase generation model for social media language, named Topic and Reference Guided Keyphrase Generation(TRGKG). TRGKG simultaneously incorporates corpus-level topics and post-level reference knowledge from retrieved posts to enrich scarce posts features. We use the neural topic model to exploit latent topics, using heterogeneous graphs inspired by [19] to obtain explicit knowledge in reference posts that contains semantic information and potential knowledge. Then both post-level reference knowledge and corpus-level latent topics are used to guide the decoding process by introducing a new topic-aware hierarchical attention and copy mechanism, which directly copies appropriate words from both the source post and its references. Besides, we combine Seq2Seq and neural topic model in a joint way so that they can benefit from each other. In detail, the proposed model share the sequence encoder in Seq2Seq with the neural topic model.

Our contributions are as follows: (1). We propose TRGKG, a keyphrase generation model using both corpus-level topics and post-level reference knowledge from retrieved posts as guidance to generate keyphrases. (2). We improve the combination method of neural topic model and Seq2Seq so that they can better promote each other. (3). Extensive experiments on two public datasets demonstrate that TRGKG achieves effective improvements over existing methods, especially on Twitter.

2 Related Work

Predicting keyphrase can be divided into keyphrase extraction and keyphrase generation. The keyphrase extraction requires that the target keyphrase must appear in the source text, and it is impossible to generate absent keyphrase. To generate absent keywords, [8] first propose a Seq2Seq architecture with copy mechanism. [1] proposes a review mechanism to consider the impact of previously generated keywords. Title concisely describes the main focus of the text.

To leverage such structure, [2] uses the title to distinguish the core information in the text. For evaluation, the above model generates multiple keyphrases using beam search decoding with fixed beam size. [21] proposed catSeq with ONE2SEQ paradigm, predicting keyphrases as sequences. However, a predefined order will introduce wrong bias during training. So [20] proposed an ONE2SET paradigm to predict the keyphrases as a set, which eliminates the bias caused by the predefined order in the ONE2SEQ paradigm. Because the ONE2ONE paradigm is known to be better for predicting absent keyphrase [7] and as absent keyphrases are frequently observed in insufficient text social media, our model follows the ONE2ONE paradigm.

For social media keyphrase prediction, target keyphrase appears in neither the target posts nor the given candidate list. [18] proposed a framework of hashtag generation of microblog, which enriches source context with its response post in external data. [5] construct structured posts by augmenting posts with missing keywords as post-level reference knowledge. [17] leverage corpus-level latent topic representations, which can be learned without requiring external data. Because of its fixed and small topic number, it is not able to distinguish different posts with similar topics. [19] retrieves articles not keywords as references because keywords lack semantic information. And it proposed to construct heterogeneous graphs using articles, keywords, reference articles, and use hierarchical attention and copy mechanisms for decoding. Different from the above methods, we use both corpus-level latent topics and post-level reference knowledge to guide keyphrase generation.

3 Methodology

The overall architecture of TRGKG is illustrated in Fig. 1. Given a source post, we firstly retrieve related posts with their keywords from the predefined index(training set) by a retriever. And we regard these related posts as reference posts. Then, source post and reference posts are encoded by sequence encoder. We build a heterogeneous graph using source post and reference posts with keywords to get the enhanced representation of source post which is regarded as the initial hidden state in decoding. Besides, we utilize a contextual neural topic model to exploit latent topics of source posts. Finally, to improve performance, we leverage both corpus-level latent topics and post-level reference knowledge in a topic-reference-aware decoder.

3.1 Retriever

Given a post x, we get related posts from the training corpus by a dense retriever. We represent the source post and all the reference posts as dense vectors. Then we calculate the cosine similarity between the source post and all reference posts to get the most similar B reference posts $X^r = \{x_1^r, x_2^r, \ldots, x_B^r\}$.

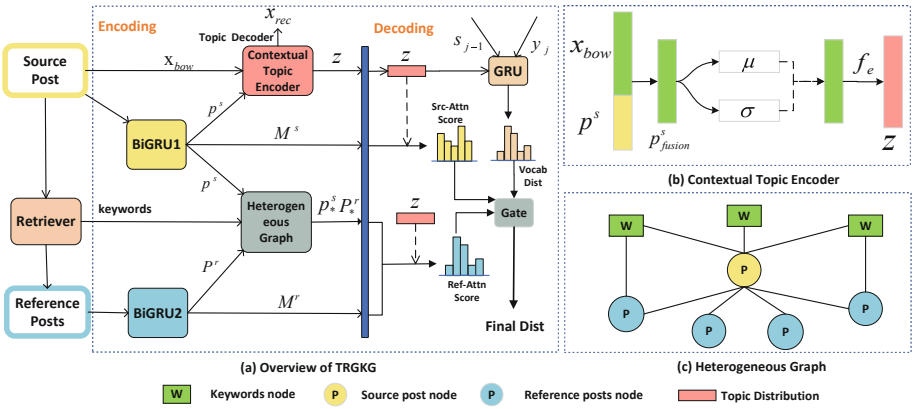


Fig. 1. Graphical illustration of our proposed TRGKG. (a) An overview of our model including encoding and decoding. For the right, we show the structure of contextual topic encoder in topic model (b) and heterogeneous graph in (c).

3.2 Encoder with Heterogeneous Graph

Sequence Encoder. We use two bidirectional gated recurrent unit (Bi-GRU) [3]: one named source sequence encoder for x, another named reference sequence encoder reference posts for $X^r = \{x_1^r, x_2^r, ..., x_B^r\}$. For source post x, firstly, each word gets word embedding vector after looking up table $\mathbf{W}_{emb}$, then using Bi-GRU to get the hidden state representations of each word in the original post $\mathbf{M}^s = \left\{\mathbf{h}_1^s, \mathbf{h}_2^s, ..., \mathbf{h}_{L_s}^s\right\}$, L_s is the number of words in source post. And we take $\mathbf{h}_{L_s}$ as the representation of source post $\mathbf{p}^s$. Similarly, using reference sequence encoder encodes B reference posts to get $\mathbf{M}^r = \{\mathbf{M}^{r_1}, \mathbf{M}^{r_2}, ..., \mathbf{M}^{r_B}\}$, here $\mathbf{M}^{r_i} = \left\{\mathbf{h}_1^{r_i}, \mathbf{h}_2^{r_i}, ..., \mathbf{h}_{L_{r_i}}^{r_i}\right\}$, and getting the representation of B reference posts $\mathbf{P}^r = \{\mathbf{p}^{r_1}, \mathbf{p}^{r_2}, ..., \mathbf{p}^{r_B}\}$.

Heterogeneous Graph. The heterogeneous graph is used to enhance the feature representation of posts. Below we introduce the heterogeneous graph module through three aspects: constructing the heterogeneous graph, initializing graph, and graph updating.

Muti-post as Heterogeneous Graph. Given a graph, $G = \{V, E\}$, V represents nodes, E represents edges between nodes. Our undirected heterogeneous graph can be formally defined as $V = V_p \cup V_w$, and $E = E_{w2p} \cup E_{p2p}$. Here, $V_p = x \cup X^r$ corresponds to source post node and B reference post nodes. $V_w = \{w_1, .., w_M\}$ means M unique key words retrieved by TF-IDF from source post and reference posts, $E_{p2p} = \{e_1, ..., e_b, ..., e_B\}$, here e_b means edge weight between b-th reference post and source post, $E_{w2p} = \{e_{1,1}, ..., e_{i,j}, ..., e_{M,B+1}\}$, here $e_{i,j}$ means edge weight between i-th keywords and j-th post.

Graph Initializers. For each post node, we apply $\mathbf{p}^s$ and $\mathbf{P}^r$ as the initial feature of node. For the keyword nodes, we apply word embedding as its initial node $\mathbf{w}_i = \mathbf{W}_{emb}(w_i)$. There are two types of edges in heterogeneous graph(i.e. post-to-post edge and word-to-post edge). To include information about the importance of relationships, we infuse cosine similarity between source post and reference posts in the edge weights of E_{p2p}. To include information about the significance of the relationships between keyword and document nodes, we infuse TF-IDF values in the edge weights of E_{w2p}.

Graph Aggregating and Updating. Given a constructed graph G with node features $\mathbf{p}^s$ and $\mathbf{P}^\mathbf{r}$ and edge features $\mathbf{E}$, Graph Attention networks(GAT) [16] is used to update the representations of each node in graph. We donate the node hidden state of input node as $\mathbf{l}_i$, $i \in \{1, ..., M + B + 1\}$. With the additional edge feature, the graph attention layer is designed as follows:

$$z_{ij} = \text{LeakyReLU}\left(\mathbf{W}_a \left[\mathbf{W}_q \mathbf{l}_i; \mathbf{W}_k \mathbf{l}_j; \mathbf{e}_{ij}\right]\right) \tag{1}$$

$$\alpha_{ij} = \frac{\exp\left(z_{ij}\right)}{\sum_{l \in \mathcal{N}_i} \exp\left(z_{il}\right)} \qquad \mathbf{u}_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}_v \mathbf{l}_j\right) \tag{2}$$

$\mathbf{W}_a, \mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ are the trainable weights, $\mathbf{e}_{ij}$ is the embedding of edge feature and α_{ij} is the attention score between $\mathbf{l}_i$ and $\mathbf{l}_j$. We use muti-head attention is defined as follows, where $\|$ means the operation of concatenation:

$$\mathbf{u}_i = \|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{l}_j\right) \tag{3}$$

After each attention layer, we introduce a residual connection and position-wise feed-forward (FFN) layer just as Transformer [15]. To transmit the messages between source post node, references nodes, and keywords nodes, after the initialization, we update each type of nodes separately as follows:

$$\begin{aligned}
\mathbf{L}_w^1 &= \text{FFN}\left(\text{GAT}\left(\mathbf{L}_w^0, \mathbf{L}_p^0, \mathbf{L}_p^0, \mathbf{E}_{w2p}\right) + \mathbf{L}_w^0\right) \\
\mathbf{L}_p^1 &= \text{FFN}\left(\text{GAT}\left(\mathbf{L}_p^0, \mathbf{L}_w^1, \mathbf{L}_w^1, \mathbf{E}_{w2p}\right) + \mathbf{L}_p^0\right) \\
\mathbf{L}_p^2 &= \text{FFN}\left(\text{GAT}\left(\mathbf{L}_p^1, \mathbf{L}_p^1, \mathbf{L}_p^1, \mathbf{E}_{p2p}\right) + \mathbf{L}_p^1\right)
\end{aligned} \tag{4}$$

where $\mathbf{L}_w \cup \mathbf{L}_p$ are the node features, $\mathbf{E}_{p2p} \cup \mathbf{E}_{w2p}$ are the edge features. Firstly, we aggregate post-level information to update word nodes, then we aggregate updated word nodes to update the post nodes, finally post nodes updated again by the updated post nodes. Each iteration contains a post-to-word, a word-to-post and a post-to-post update process. The above process is performed I round to get the enhanced post representation $\mathbf{L}_p^I$. We separate $\mathbf{p}_*^s$ and $\mathbf{P}_*^r = \{\mathbf{p}_*^{r_1}, \mathbf{p}_*^{r_2}, ..., \mathbf{p}_*^{r_B}\}$ from $\mathbf{L}_p^I$ as representation of source post and each reference. When the encoder finished, we use $\mathbf{p}_*^s$, $\mathbf{P}_*^r$, $\mathbf{M}^s$ and $\mathbf{M}^r$ to guide keyphrase generation.

3.3 Contextual Neural Topic Model

Neural topic model is based on variational autoencoder [6] framework, discovering latent topics via an encoding-decoding process. Its input is a V-dimensional BoW vector of source post $\mathbf{x}_{bow}$. And we depress it into a K-dimensional latent topic distribution $\mathbf{z}$ by a topic encoder. Then the topic decoder conditioned on $\mathbf{z}$, tries to reconstruct the BoW vector of x and outputs $\mathbf{x}_{rec}$. V is the vocabulary size after removing the stopwords, K is the number of topics. In particular, the decoder simulates the generation process of the topic model.

Topic Encoder. $q_\phi(\mathbf{g}|\mathbf{x}_{bow})$, it generates $\mu = f_\mu(\mathbf{x}_{bow}), \sigma = f_\sigma(\mathbf{x}_{bow})$, then obtain $\mathbf{g} = \mu + \sigma \cdot \epsilon$. It learn the latent topic distribution $\mathbf{z} = f_e(\mathbf{g})$, where f_μ and f_σ are linear layers with ReLU activation, f_e is a ReLU activation and a Gussian softmax [10].

Topic Decoder. $p_\theta(\mathbf{x}_{rec}|\mathbf{g})$, transforms g into predicted probability of each word in vocabulary by using ReLU-activated linear layers f_d:

$$\mathbf{x}_{rec} = \text{softmax}(f_d(\mathbf{z})) \tag{5}$$

The neural topic model above only uses the BoW vector as input. But BoW vector has some limitations, such as ignoring sequence information of the text. Fortunately, the sequence encoder in the encoder can capture post semantics better, so we use the representation of source post $\mathbf{p}^s$ with sequence information as an input of the topic model. Then, the topic model can utilize the BoW vector and contextual representation to exploit latent topics. We call it contextual neural topic model, because it shares the sequence encoder in Seq2Seq.

Contextual Topic Encoder. It can be regarded as a two-step topic encoding process. To keep valuable information in both contextual representation and BoW representation, we first fuse $\mathbf{p}^s$ and $\mathbf{x}_{bow}$ as follows:

$$\mathbf{p}^s_{fusion} = \mathbf{W}_f[\mathbf{x}_{bow}; \mathbf{p}^s] + \mathbf{b}_f \tag{6}$$

$\mathbf{W}_f$ and $\mathbf{b}_f$ are trainable parameters, $[\mathbf{x}_{bow}; \mathbf{p}^s]$ means the concatenation of $\mathbf{x}_{bow}$ and $\mathbf{p}^s$. $\mathbf{p}^s_{fusion}$ is a V-dimensional vector. And Then we use the same method in topic encoder to get topic distribution $\mathbf{z}$:

$$\mu = f_\mu(\mathbf{p}^s_{fusion}) \qquad \sigma = f_\sigma(\mathbf{p}^s_{fusion}) \tag{7}$$

$$\mathbf{g} = \mu + \sigma \cdot \epsilon \tag{8}$$

$$\mathbf{z} = f_e(\mathbf{g}) \tag{9}$$

For decoder, we use the topic decoder above to reconstruct the BoW vector x_{rec}. In the following, we adapt latent topic distribution $\mathbf{z}$ as the representation of source post and use it to guide generating keyphrases.

3.4 Topic-Reference-Aware Decoder

Sequence Decoder. We use $\mathbf{p}_*^s$ as the initial state of GRU decoder. Besides the general hidden state update, we also take the topic distribution into consideration. $\mathbf{s}_{j-1}$ is the previous step hidden state. The j-th decode process is described as follows:

$$\mathbf{s}_j = \mathrm{GRU}\left(\left[\mathbf{W}_{emb}\left(y_j\right); \mathbf{z}\right], \mathbf{s}_{j-1}\right) \tag{10}$$

Topic-Aware Hierarchical Attention. To determine important information from source and reference posts, we use new hierarchical attention with latent topics. Because latent topics enable the model to focus on some topic words in source and reference posts. On the other hand, latent topics can correct irrelevant information in reference posts due to retrieval. We generalize topic-aware hierarchical attention with the following formula:

$$\mathbf{c}_j = \mathrm{TAHierAttn}\left(\mathbf{s}_j, \mathbf{M}^s, \mathbf{M}^r, \mathbf{P}_*^r, \mathbf{z}\right) \tag{11}$$

The context vector $\mathbf{c}_j$ is obtained by topic-aware hierarchical attention. $\mathbf{c}_j$ is computed as follows:

$$\mathbf{c}_j^s = \sum_{i=1}^{L_s} \alpha_{ji}^s \mathbf{h}_i^s, \qquad \mathbf{c}_j^r = \sum_{i=1}^{B} \sum_{k=1}^{L_{r_i}} \alpha_{ji}^r \alpha_{jk}^{r_i} \mathbf{h}_k^{r_i} \tag{12}$$

$$\theta = \mathrm{sigmoid}\left(\mathbf{W}_\theta[\mathbf{c}_j^s; \mathbf{c}_j^r] + \mathbf{b}_\theta\right), \qquad \mathbf{c}_j = \theta \cdot \mathbf{c}_j^s + (1-\theta) \cdot \mathbf{c}_j^r \tag{13}$$

where $\mathbf{c}_j^s$ is context vector from source post and $\mathbf{c}_j^r$ is context vector from reference posts. $\mathbf{W}_\theta$ and $\mathbf{b}_\theta$ are trainable parameters, $\mathbf{h}_i^s \in \mathbf{M}^s$ and $\mathbf{h}_k^{r_i} \in \mathbf{M}^{r_i}$, θ is a gate for two context vectors. α^s is a word-level attention scores over the source post $\mathbf{M}^s$, α^r is a post-level attention scores over the reference posts $\mathbf{P}_*^r$ and α^{r_i} is a word-level attention scores over the i-th reference post $\mathbf{M}^{r_i}$.

We integrate the topic distribution when calculating the attention scores of source post and reference posts. Here's how they are calculated:

$$\alpha_{ji}^s = \frac{\exp\left(f_\alpha\left(\mathbf{h}_i^s, \mathbf{s}_j, \mathbf{z}\right)\right)}{\sum_{i'=1}^{L_x} \exp\left(f_\alpha\left(\mathbf{h}_{i'}^s, \mathbf{s}_j, \mathbf{z}\right)\right)} \tag{14}$$

$$\alpha_{ji}^r = \frac{\exp\left(f_\alpha\left(\mathbf{p}_*^{r_i}, \mathbf{s}_j, \mathbf{z}\right)\right)}{\sum_{i'=1}^{B} \exp\left(f_\alpha\left(\mathbf{p}_*^{r_{i'}}, \mathbf{s}_j, \mathbf{z}\right)\right)} \tag{15}$$

$$\alpha_{jk}^{r_i} = \frac{\exp\left(f_\alpha\left(\mathbf{h}_k^{r_i}, \mathbf{s}_j, \mathbf{z}\right)\right)}{\sum_{i'=1}^{L_{r_i}} \exp\left(f_\alpha\left(\mathbf{h}_{i'}^{r_i}, \mathbf{s}_j, \mathbf{z}\right)\right)} \tag{16}$$

Equation 14 calculates the attention scores of source posts. Equation 15 and Eq. 16 calculate the attention score of reference posts. f_α is defined as follows, $\mathbf{v}_\alpha, \mathbf{W}_\alpha$ and $\mathbf{b}_\alpha$ are trainable parameters:

$$f_\alpha\left(\mathbf{h}_i^s, \mathbf{s}_j, z\right) = \mathbf{v}_\alpha^T \tanh\left(\mathbf{W}_\alpha\left[\mathbf{h}_i^s; \mathbf{s}_j; z\right] + \mathbf{b}_\alpha\right) \tag{17}$$

Final Scores with Copy Mechanism. Firstly the j-th target generation scores on predefined vocabulary p_{gen} are computed by:

$$p_{gen} = \text{softmax}\left(\mathbf{W}_{gen}\left[\mathbf{s}_j; \mathbf{c}_j; \mathbf{z}\right] + \mathbf{b}_{gen}\right) \tag{18}$$

We define the source scores on post's words p_{sou}, the reference scores p_{ref} on reference posts' words. The finial scores p_j is computed as a combination of three scores using soft gate $\lambda \in \mathbb{R}^3$, determining the importance of three scores:

$$\lambda = \text{softmax}\left(\mathbf{W}_{\lambda}\left[\mathbf{s}_j; \mathbf{c}_t; \mathbf{z}; \mathbf{W}_{emb}(y_{t-1})\right] + \mathbf{b}_{\lambda}\right) \tag{19}$$

$$p_j = \lambda_1 p_{gen} + \lambda_2 p_{sou} + \lambda_3 p_{ref} \tag{20}$$

where source scores $p_{sou} = \sum_i^{L_s} \alpha_{ji}^s$ are the copy probability from source post, reference scores $p_{ref} = \sum_i^B \sum_k^{L_{r_i}} \alpha_{jk}^{r_i}$ are the copy probability from all the reference posts. We adopt copy mechanism above, which allows keywords to be directly extracted from the source input. We define a post generate its keyphrase y with probability:

$$\Pr(\mathbf{y} \mid \mathbf{x}) = \prod_{j=1}^{|\mathbf{y}|} \Pr\left(y_j \mid \mathbf{y}_{1:j-1}, \mathbf{M}^s, \mathbf{M}^r, \mathbf{P}_*^r, \mathbf{z}\right) \tag{21}$$

where $|\mathbf{y}|$ is the length of the keyphrase, $\Pr\left(y_j \mid \mathbf{y}_{1:j-1}, \mathbf{M}^s, \mathbf{M}^r, \mathbf{P}_*^r, \mathbf{z}\right)$ as a word distribution over vocabulary, denoted $\mathbf{p}_j$, means the how likely a word to fill the j-th slot in target keyphrase.

3.5 Jointly Training

We jointly train our neural network of learning topic modeling and keyphrase generation. The objective function of the topic model is defined as negative of evidence lower bound, as shown as follows:

$$\mathcal{L}_{NTM} = D_{KL}(p_\theta(\mathbf{g}) \mid q_\phi(\mathbf{g} \mid \mathbf{x})) - \mathbf{E}_{q_\phi(\mathbf{g}|\mathbf{x})}[p_\theta(\mathbf{x} \mid \mathbf{g})] \tag{22}$$

where the first term D_{KL} is Kullback-Leibler divergence loss, and the second term E reflect the reconstruction loss. $q_\phi(\mathbf{g} \mid \mathbf{x})$ and $p_\theta(\mathbf{x} \mid \mathbf{g})$ means the network of contextual topic encoder and topic decoder.

The loss of keyphrase generation is cross entropy expressed as :

$$\mathcal{L}_{KG} = -\sum_{n=1}^{N} \log\left(\Pr\left(\mathbf{y}_n \mid \mathbf{x}_n\right)\right) \tag{23}$$

The final loss of the entire network is a linear combination of two loss:

$$\mathcal{L} = \mathcal{L}_{NTM} + \gamma \cdot \mathcal{L}_{KG} \tag{24}$$

γ is a factor to balance two loss. For inference, we use beam search and then use top-ranked keyphrases as final predictions.

4 Experiment Settings

4.1 Datasets

We conduct experiments on two social media English datasets, Twitter and StackExchange [17]. In Twitter including microblog posts, 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. StackExchange is collected from a Q&A platform. For one text, it is the concatenation of a title and a description, and the hashtags are manually by the user. For model 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. Table 1 gives the details of two datasets.

Table 1. Statistics of datasets. Avg.len: average number of tokens per post. Unq.D and Unq.KP: number of distinct posts and keyphrases. Avg.KP:average number of keyphrases per post. Abs.KP(%): ratio of absent keyphrases. We report average performance with different random seeds of 5 runs.

Datasets	Avg.len	Unq.D	Avg.KP	Unq.KP	Abs.KP(%)
Twitter	19.52	44113	1.13	4347	71.35
StackExchange	87.94	49447	2.43	12114	54.32

4.2 Comparisons and Evaluation

To demonstrate the effectiveness of our method, we compared our method with keyphrase extraction and keyphrase generation models. For keyphrase extraction models, we use TF-IDF scores, TextRank algorithm [11] and keyphrase extraction method based on sequence tagging [22]. For keyphrase generation, we compared the following models: SEQ2SEQ [8] which uses Seq2Seq framework, CopyRNN [8] which based on SEQ2SEQ with copy mechanism, CorrRNN [1] which exploring keyphrases correlations, TAKG [17] using latent topic helping keyphrase generation, CopyRNN-GATER [19] using a heterogeneous graph to capture references knowledge to guide decoding process, TG-NET [2] using the struct of title and descriptions and GSEnc [5] arguments structure using other document and uses a graph to obtain structure-aware representation for the decoding process.

For evaluations, we adopted macro-average F1@K and mean average precision (MAP) as evaluation metrics. We report F1@1/3 for the twitter dataset, F1@3/5 for StackExchange, squaring up the average number of keyphrases in datasets. For two datasets, we report MAP@5 over top-5 predictions.

4.3 Implementation Details

Following the previous work, we train our network in a SEQ2ONE paradigm. We keep the parameter settings used in [17]. For the additional graph module, we set the number of reference posts B among {3, 5,10} and the number of keywords M per post among 3, 10, 20. Because of a longer text in StackExchange, B was 3 for StackExchange and 5 for Twitter, M was 10 for StackExchange and 3 for Twitter. We use dense retriever MPNet [14], which is s a pre-training method for language understanding tasks. For a fair comparison, we use the same B, M, and dense retriever in CopyRNN-GATER. In the training process, in order for the contextual neural topic model to utilize the sequence encoder in the SEQ2SEQ model, we pretrain the KG module for 2 epochs. Then since the neural topic model trains slower than the KG module, we also warm up the neural topic model several epochs, specifically, we warm up 50 epochs on Twitter and 20 epochs on StackExchange. We empirically set γ to 1. We set the beam size to 50 for the inferencing process.

Table 2. The performance of keyphrase prediction. The best results are bold. TAKG-CNTM: TAKG with Contextual Neural Topic Model.

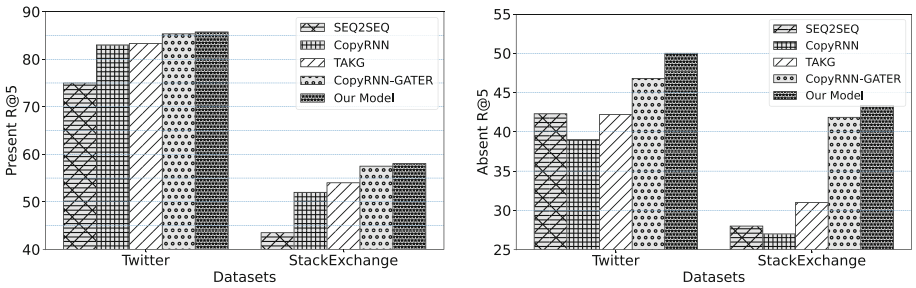
Model	Twitter			StackExchange		
	F1@1	F1@3	MAP	F1@3	F1@5	MAP
TF-IDF	1.16	1.14	1.89	13.50	12.74	12.61
TEXTRANK	1.73	1.94	1.89	6.30	8.28	4.76
SEQ-TAG	22.79	12.27	22.44	17.58	12.82	19.03
SEQ2SEQ	34.10	26.01	41.11	22.99	20.65	23.95
CopyRNN	36.60	26.79	43.12	31.53	27.41	33.45
CorrRNN	34.97	26.13	41.64	30.89	26.97	32.87
TG-NET	–	–	–	32.02	27.84	34.05
TAKG	38.49	27.84	45.12	33.41	29.16	35.52
GSEnc	38.80	28.10	45.50	35.20	30.90	37.80
CopyRNN-GATER	41.09	29.69	48.06	37.96	33.33	41.16
TRGKG	**42.20**	**30.68**	**49.69**	**38.66**	**33.76**	**41.75**
TAKG-CNTM	39.24	28.73	46.20	33.80	29.78	36.00

5 Results and Analysis

5.1 Performance of Keyphrase Generation

Table 2 shows the performance of keyphrase predicted by our model and other baselines. Keyphrase extraction models show much worse performance than keyphrase generation models on both datasets. The language style of social

media result in a large number of absent keyphrases, which is impossible for to extract from the original post. Among keyphrase generation models, latent topics are useful to prediction, TAKG with latent topics can get better result. However, GSEnc and CopyRNN-GATER achieve the best performance among all baselines. It shows that retrieved posts and keywords are beneficial to prediction of keyphrase. Our model outperform all the state-of-the-art baseline models on both datasets. Because our model incorporates not only latent topics but also reference knowledge. We replace he original topic model in TAKG with the proposed contextual neural topic model marked as TAKG-CNTM. The results show that TAKG-CNTM can achieve better performance than TAKG. This proves the effectiveness of contextual neural topic model.



(a) Present Keyphrase Prediction (b) Absent Keyphrase Prediction

Fig. 2. The prediction results of present and absent keyphrases.

5.2 Prediction of Present and Absent Keyphrase

We further discuss the performance of our model in predicting the present and absent keyphrase. The results are shown in Fig. 2. For the present keyphrase, F1@1 is adopted for evaluating. And we use recall@5 for the absent keyphrases.

An important finding is that reference knowledge from retrieved posts is more helpful than latent topics in generating not only present but also absent keyphrases. Results show that CopyRNN-GATER and our model are better at predicting two types of keyphrases than TAKG. This may be because reference knowledge provides more fine-grained post-level information than corpus-level topics. Especially, reference knowledge enables the model to capture more keywords to copy, which can greatly improve the prediction performance of absent keyphrases. However, the results indicate that our model outperforms all comparison models in predicting two types of keyphrases especially on Twitter. It is attributed to our ability to incorporate post-level reference knowledge and corpus-level topics.

5.3 Ablation Study

Topics Are Useful. To verify the efficiency of topics, we removed topics in two ways: (1). Remove topics when calculating the attention score of source post, in Eq. 14. (2). Remove topics when calculating the attention score of reference posts, in Eqs. 15 and 16. (3). Remove both of the above. The results in Table 3 show that these ablation methods make the model perform worse. This suggests that topics not only facilitate the identification of key information in source post but also the reference posts. One possible explanation is that latent topics of source post can correct irrelevant information in reference posts retrieved.

Table 3. Results of our ablation study. w/o: without, topic-sour: source attention scores with topics, topic-refer: reference attention scores with topics, topic-all: source and reference attention attention scores with topics, joint-train: jointly training.

Model	StackExchange F1@3	Twitter F1@1
Our model (w/o topic-sour)	38.40	41.73
Our model (w/o topic-refer)	38.23	41.87
Our model (w/o topic-all)	38.19	41.70
Our model (w/o joint-train)	38.51	**42.35**
Our full model	**38.66**	42.20

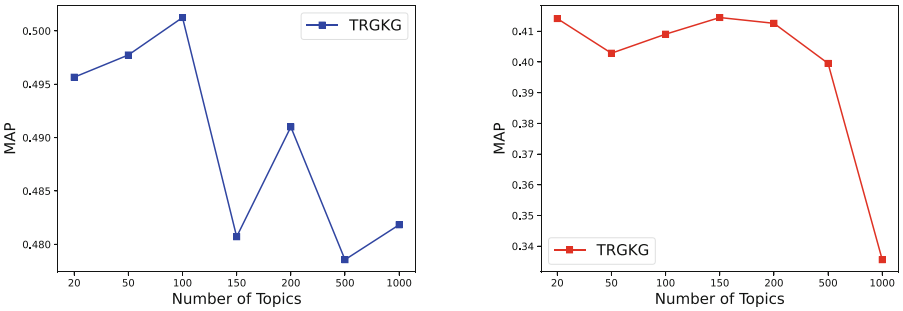
Joint or Pipeline. We train the topic model separately from the rest of the TRGKG as ablation. Surprisingly, this way of training performed almost identically to the joint training method. This is quite different from TAKG, where joint training with TAKG results in a significant improvement in model performance. This may be due to the shared sequence encoder between the neural topic module and Seq2Seq in TRGKG.

5.4 Influence of the Number of Topics

We compare the effect of different topic numbers $k \in [20]$ of keyphrase generation performance, and the results are shown in Fig. 3. On Twitter, the number of topics has little effect on the overall performance, with a maximum difference of about 2%. On the StackExchange, the model performance shows a trend of increasing first and then decreasing when the number of topics is between [20]. Unlike Twitter, the performance of keyword generation starts to drop dramatically when the number of topics is set above 200. This may be due to the different data conditions of the datasets.

5.5 Case Study

We further present a case study, where the top five outputs of some comparison models have displayed in Table 4. As can be seen, only our model successfully generates the golden present keyphrase '*love*' and absent keyphrase '*quote*'. On the contrary, TAKG not only fails to generate the absent keyphrase '*quote*' but also generates irrelevant keyphrases. It is possible that the words '*game*', '*win*' and '*play*' in the posts are causing the model to identify the wrong topic '*super bowl*'. CopyRNN-GATER fails to generate the present keyphrase '*love*', missing the understanding of the meaning of the original post. This demonstrates that using both the corpus-level topics and post-level reference knowledge can generate more accurate absent and present keyphrases compared to the baselines.



(a) MAP on Twitter (b) MAP on StackExchange

Fig. 3. Performance for our model with different topic numbers.

Table 4. A keyphrase prediction example of different models. Bolded and italic indicate the golden keyphrases.

Posts	Love is a game that two can play and both win
Golden Keyphrases	***love***, ***quote***
TAKG	***love***, super bowl, sb45, packers, steelers
CopyRNN-GATER	oh teen quotes, just saying, ***quote***, daily teen, omgsotrue
TRGKG	***love***, real talk, ***quote***, fact, omgsotrue

6 Conclusion

We proposed a keyphrase generation model named TRGKG that incorporates both corpus-level latent topics and post-level reference knowledge. We exploited latent topics using a contextual neural topic model and capture knowledge from reference posts based on the heterogeneous graph. And they are fused together by

a topic-aware hierarchical attention to guide the decoding process. Experimental results show that our method outperforms all state-of-the-art models on Twitter and StackExchange.

Acknowledgements. This work is supported by the National Key R&D Program of China under Grants (No. 2018YFB0204300).

References

1. Chen, J., Zhang, X., Wu, Y., Yan, Z., Li, Z.: Keyphrase generation with correlation constraints. In: Empirical Methods in Natural Language Processing (2018)
2. Chen, W., Gao, Y., Zhang, J., King, I., Lyu, M.R.: Title-guided encoding for keyphrase generation. In: National Conference on Artificial Intelligence (2019)
3. Cho, K., van Merriënboer, B., Gulcehre, C.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP (2014)
4. He, Q., Yang, J., Shi, B.: Constructing knowledge graph for social networks in a deep and holistic way. In: Companion Proceedings of the Web Conference (2020)
5. Kim, J., Jeong, M., Choi, S., won Hwang, S.: Structure-augmented keyphrase generation. In: Empirical Methods in Natural Language Processing (2021)
6. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: International Conference on Learning Representations (2014)
7. Meng, R., Yuan, X., Wang, T., Zhao, S., Trischler, A., He, D.: An empirical study on neural keyphrase generation. In: NAACL (2021)
8. Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., Chi, Y.: Deep keyphrase generation. In: Meeting of the Association for Computational Linguistics (2017)
9. Meng, X., Wei, F., Liu, X., Zhou, M., Li, S., Wang, H.: Entity-centric topic-oriented opinion summarization in twitter. In: Knowledge Discovery and Data Mining (2012)
10. Miao, Y., Grefenstette, E., Blunsom, P.: Discovering discrete latent topics with neural variational inference. Computation and Language. arXiv preprint arXiv:1706.00359 (2017)
11. Mihalcea, R., Tarau, P.: Textrank: Bringing order into text. In: Empirical Methods in Natural Language Processing (2004)
12. Panwar, M., Shailabh, S., Aggarwal, M., Krishnamurthy, B.: Tan-ntm: topic attention networks for neural topic modeling. In: ACL (2021)
13. Ruths, D., Pfeffer, J.: Social media for large studies of behavior. Science 1063–1064 (2014)
14. Song, K., Tan, X., Qin, T., Lu, J., Liu, T.Y.: Mpnet: masked and permuted pre-training for language understanding. Computation and Language. arXiv preprint arXiv:2004.09297 (2020)
15. Vaswani, A., et al.: Attention is all you need. In: Neural Information Processing Systems (2017)
16. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. STAT 20 (2017)
17. Wang, Y., Li, J., Chan, H.P., King, I., Lyu, M.R., Shi, S.: Topic-aware neural keyphrase generation for social media language. In: ACL (2019)
18. Wang, Y., Li, J., King, I., Lyu, M.R., Shi, S.: Microblog hashtag generation via encoding conversation contexts. In: North American Chapter of the Association for Computational Linguistics (2019)

19. Ye, J., Cai, R., Gui, T., Zhang, Q.: Heterogeneous graph neural networks for keyphrase generation. Computation and Language. arXiv preprint arXiv:2109.04703 (2021)
20. Ye, J., Gui, T., Luo, Y., Xu, Y., Zhang, Q.: One2set: generating diverse keyphrases as a set. In: Meeting of the Association for Computational Linguistics (2021)
21. Yuan, X., et al.: One size does not fit all: Generating and evaluating variable number of keyphrases. In: Meeting of the Association for Computational Linguistics (2020)
22. Zhang, Q., Wang, Y., Gong, Y., Huang, X.: Keyphrase extraction using deep recurrent neural networks on twitter. In: EMNLP (2016)