Knowledge-Based Systems 243 (2022) 108488

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

# GCN-based document representation for keyphrase generation enhanced by maximizing mutual information

Peng Yang [a,b,c,*], Yanyan Ge [a,c], Yu Yao [a,c], Ying Yang [a,c]

[a] *School of Computer Science and Engineering, Southeast University, Nanjing, China*
[b] *School of Cyber Science and Engineering, Southeast University, Nanjing, China*
[c] *Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing, China*

A R T I C L E   I N F O

*Article history:*
Received 22 March 2021
Received in revised form 18 February 2022
Accepted 19 February 2022
Available online 25 February 2022

*Keywords:*
Keyphrase generation
Sequence-to-sequence
Graph convolutional network
Mutual information

A B S T R A C T

Keyphrase generation is an important fundamental task of natural language processing, which can help users quickly obtain valuable information from a large number of documents especially when they are facing with informal social media text. Existing Recurrent Neural Network (RNN) based keyphrase generation approaches cannot properly model the dependency structure of the informal text, which is often implicit between those distant words and plays an important role in extracting salient information. To obtain core features of text, we apply Graph Convolutional Network (GCN) on document-level graph to capture dependency structure information. The GCN-based node representations are further fed into a predictor network to provide potential candidates for copying mechanism. Moreover, we utilize a novel variational selector network to determine the final selection probability of each word in a phrase, which relies on its probabilities of copying from a given document and being generated from a vocabulary. Eventually, we introduce an enhancement mechanism to maximize the mutual information between document and generated keyphrase, thus ensuring the consistency between them. Experiment results show that our model outperforms previous state-of-the-art baselines on three social datasets, including Weibo, Twitter and StackExchange.

## 1. Introduction

Keyphrase Generation (KG) is one of the most fundamental and attractive tasks in natural language processing (NLP). Keyphrase generation concerns predicting a set of phrases or words that summarize the core and important information in the given document. As keyphrases provide brief but representative information, they can be applied to a wide range of downstream NLP tasks, such as text summarization [1], opinion mining [2,3], and question generation [4].

Recently, Sequence-to-Sequence (Seq2Seq) [5]-based methods have achieved promising performance on text generation tasks. The simplest Seq2Seq model consists of an RNN-based encoder and decoder. Meng et al. [6] first proposed CopyRNN, a Seq2Seq model that incorporates a copying mechanism [7], to solve the KG task. Then, several extensions of CopyRNN were proposed [8–10]. These Seq2Seq based methods overcome the major drawback of extraction methods, which is that they cannot yield absent words (i.e., words do not exist in the source document). However, most existing research efforts on keyphrase generation have been focused on scientific articles, where the information flow is very clear throughout the paragraphs or sections. Different from formal documents, social media text is often casual and colloquial, and its salient information is scattered, making it difficult for current KG models to focus on the crucial parts of the text. Purely RNN-based methods can better model the dependencies between adjacent words but cannot model well those dependency structures with distance spanning. And the nature of those information is implicit and across words rather than explicit. To alleviate the information scatter issue, it is necessary to design and implement new KG methods instead of directly applying the current generation models. Wang et al. [10] assembled a neural topic model (NTM) in CopyRNN, namely TAKG, which leverages latent topics to enrich useful features. However, when TAKG is based on the topic model, it primarily faces the practical problem of fixed topic distribution, and the extracted topics are not explicit enough.

In this work, we aim to tackle the information scatter challenge by combining the dependency information obtained by a graph convolution network (GCN) [11] with contextual information. Specifically, for each text, we first construct a syntactic graph for each sentence and then build the edges between the sentences according to predefined rules to obtain a document-level graph. Each node in the graph corresponds to an individual word, and the edge between a pair of nodes represents their the

[*] Corresponding author at: School of Computer Science and Engineering, Southeast University, Nanjing, China.
*E-mail address:* pengyang@seu.edu.cn (P. Yang).

https://doi.org/10.1016/j.knosys.2022.108488
0950-7051/© 2022 Elsevier B.V. All rights reserved.

dependency. After the graph is constructed, we input it into a GCN to propagate the contextual information among non-local words and capture the dependency structure information in the text. Intuitively, dependency information plays an important role in extracting salient features. We surmise that integrating the dependency information into the contextual information obtained by an RNN can alleviate the information scatter issue. The GCN-based node representations are further input into a predictor network to filter out noise words and provide potential candidates for the copying mechanism.

The second challenge is that existing methods need to accommodate two mechanisms: copying words from source documents, and generating semantically related words from the vocabulary [12]. Most of the recent works utilize a deterministic network to determine whether to copy a word from source as a target word. Instead, our model resorts to a novel variational selector network to determine the final selection probability for each word in a keyphrase by using amortized variational inference [13]. Using the variational mechanism, our model can model the ambiguity inherent in sources, and generate diverse words that focus on different parts of a context one at a time.

Another crucial challenge of the KG task is to ensure consistency between the source and its target keyphrase, as they should be semantically dependent on each other. Similar to Seq2Seq-based text summarization, KG models are usually trained by maximizing likelihood estimation. Such a training strategy is highly inadequate for solving the KG task. Since models cannot ensure consistency between a document and its corresponding keyphrase, they tend to generate high-frequency phrases that are not related to the source [14]. To tackle the consistency issue between documents and their generated keyphrases, we introduce an enhancement mechanism based on a mutual information maximization [15–17] strategy. Maximizing Mutual Information (MMI) learning objective can encourage the decoder to learn more unique information from the input document than other documents, so that the model can learn more semantically relevant representations of the document and phrases.

Combining the above mechanisms, we propose a novel KG model, which we refer to as **G**raph-based **K**eyphrase **G**eneration (**GKG**). We experimentally validate the GKG model on three social datasets, Twitter, Weibo and StackExchage. The empirical evaluation shows that our model significantly outperforms state-of-the-art baselines on keyphrase generation task.

To summarize, our contributions are:

- Focusing on the information scatter issue that cannot be well solved by existing RNN-based methods. The innovation of GKG is that it could model distance spanning text dependency information, which is not available with existing RNN-based methods.
- We utilize a variational selector network to determine the final selection probability for each word in a phrase, which relies on the probabilities of copying from a given document and being generated from a vocabulary.
- We introduce an enhancement mechanism based on mutual information maximization for learning the representations from documents and phrases, thus enforcing the consistency between them.
- We conduct experiments on three social datasets, including Weibo, Twitter and StackExchange. Extensive experimental results verify the effectiveness of GKG on keyphrase generation task.

## 2. Related work

### 2.1. Keyphrase generation

Keyphrase generation aims to condense a given document into a list of phrases that summarize the input's core information. Most previous KG approaches are extractive; they extract salient multi-word units from documents based on a two-step strategy [18]. Specifically, models first select a set of candidate phrases using predefined rules or templates [19–21]. Afterwards, the candidates are reranked according to supervised methods [22,23] or unsupervised algorithms [20,24–26] and the model returns the top-N phrases as keyphrases. Additionally, Zhang et al. [27] and Luan et al. [28] constructed sequence tagging models to identify keyphrases. However, extractive approaches ignore the holistic meaning of the documents, and lack the ability to produce phrases that do not appear in the source text.

Compared with extractive approaches, abstractive models are able to predict both present and absent keyphrases. Meng et al. [6] proposed CopyRNN which models keyphrase generation as a Seq2Seq task. A classical architecture for Seq2Seq models is encoder–decoder which uses an "encoder" to encode the input document into a hidden representation, and then uses a "decoder" to generate target phrases. Recently, many improvements to CopyRNN have been proposed: designing a coverage mechanism to alleviate the problem of phrase overlapping [8], implementing a review mechanism to solve the repletion problem and increase keyphrase diversity [9], introducing an adaptive reward function to overcome the limitation that CopyRNN generates insufficient keyphrases [29], applying a multi-task learning framework and integrating syntactic constraints for keyphrases to improve model performance [30]. Recent abstractive KG methods have demonstrated success in the scientific domain. However, not enough attention has been paid to addressing social media keyphrase generation, and extractive methods are widely employed. In contrast, our work focuses on the challenge of generating keyphrases using social media datasets.

Most recently, Wang et al. [10] proposed a topic-aware model, namely, TAKG, that utilizes the corpus-level latent topic representations to enrich the context representation of the input obtained by the encoder. Our work is partially inspired by TAKG, but our approach is quite different from theirs. We propose utilizing the dependency structure information from documents, which can be captured by a GCN. It is potentially useful to alleviate the information scatter issue by combining dependency information with contextual information.

### 2.2. Application of graph convolutional networks in NLP

The graph neural network (GNN) has achieved competitive results for representation learning, and is effective at processing graph structure data. GNN and its derived variants generally research how to update a node's representation by aggregating the representations of its neighbors [11,31–35]. Our framework utilizes the recently proposed GCN to learn word and document representations, which defines the convolution using the first-order graph Laplacian methods [11]. Some recent studies have been applied the GNN to the NLP task, which involves finding strategies for modeling non-structural text as graph. Peng et al. [36], Yao et al. [37], and Zhang et al. [38] proposed converting a document into a word co-occurrence graph based on point-wise mutual information. Other works attempt to transform the dependency tree into a graph to integrate syntactic structure information [4,39]. In addition, several studies [40–42] construct entity graphs that utilize the entities from scattered paragraphs and predefined connection rules.

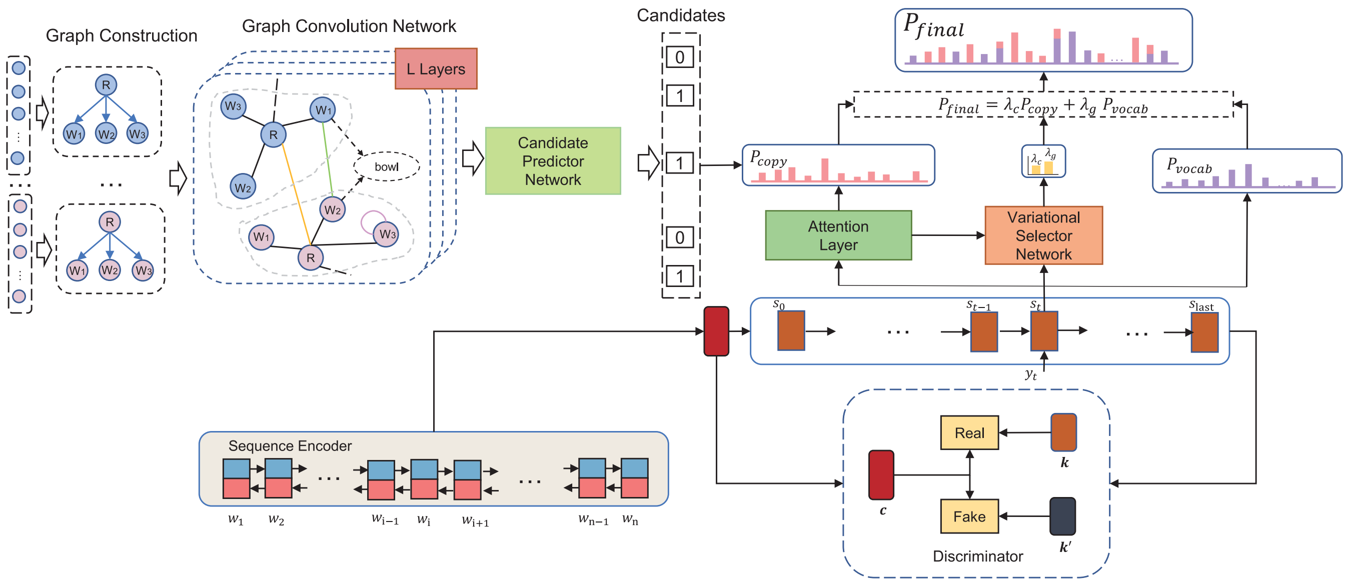

**Fig. 1.** Model architecture. The model consists of three parts: (1) A sequence decoder to encoder the input text (top right corner). (2) A graph encoder to embed the document-level graph via GCN (bottom right corner), We omit several edges, such as self-looping edges of all words, for brevity. Different colors represent different types of edges in the document-graph. (3) A sequence decoder to generate keyphrase from document representation (left). In discriminator part, Orange and dark green rectangles denote representations of target phrase (positive example) and another phrase (negative example) in a batch, respectively.

Unlike the above methods, we not only consider the syntactic structure to obtain intra-sentence dependency information, but also construct the edges between the sentences to transfer information. Then, we apply the GCN over the graph and use a predictor network to predict potential clue words for the copying mechanism together with a Gumbel-Softmax estimator.

### *2.3. Mutual information for representation learning*

One of the most important objectives of deep learning is to obtain meaningful representations. Several recent studies inspired by the infomax principle [43] have learned representations by maximizing the MI between the model's input and output, and have shown promising empirical results in representational learning in different domains, including images [16], graphs [15,17,44,45], and text [14,46,47]. Motivated by DIM [16] for image representation learning tasks, with an unsupervised learning objective aims to maximize the MI between the global representation and local context features. Different from DIM, our model primarily works with sequence generation using the Seq2Seq model. We introduce an enhancement mechanism to enforce consistency between the document and its target keyphrase withs by maximizing a lower bound on the MI between them.

## 3. Model architecture

In keyphrase generation task, given a social media document consisting of a sequence of n tokens $P = (x_1, x_2, \ldots, x_n)$, the target output is a phrase containing m words $T = (y_1, y_2, \ldots, y_m)$. To this end, we opt for the paradigm adopted by Meng et al. [6] for training model. When data preprocessing, we divide each one2many data (example contains one document and multiple target phrases) into one2one examples.

The architecture of our model is depicted in Fig. 1, which is composed of a sequence-encoder based on RNN, a graph-encoder and a sequence-decoder. The model first performs sequence encoder to get contextual representations of document and words. After that, it applies graph-encoder onto document-level graph to predict whether a word will be retained or not and get structure feature of document. Finally, the sequence-decoder takes the document representation as input, and generates phrase.

### *3.1. Sequence encoder*

Given a document consisting of a sequence of n words, we map each word and its Part-of-Speech (POS) into low-dimensional real-valued vectors, $\boldsymbol{w}_i$, $\boldsymbol{d}_i$, respectively. Then, we concatenate the word and POS embeddings to form the final word feature, $\boldsymbol{x}_i = [\boldsymbol{w}_i; \boldsymbol{d}_i]$. After obtaining the word feature, we apply a bidirectional Gated Recurrent Unit (Bi-GRU) to produce hidden state vectors $\boldsymbol{H}^s = \{\boldsymbol{h}_1^s, \boldsymbol{h}_2^s, \ldots, \boldsymbol{h}_t^s, \ldots, \boldsymbol{h}_n^s\}$, where $\boldsymbol{h}^s = \{\overrightarrow{\boldsymbol{h}_t^s}; \overleftarrow{\boldsymbol{h}_t^s}\}$ is the concatenation of forward and backward hidden states at time step $t$, and $\boldsymbol{c} = \boldsymbol{h}_n^s$ is used as document's representation.

### *3.2. Graph encoder*

In this work, we obtain dependency features by applying multi-layer graph convolution over the document-level graph of the document, and use a candidate predictor network to provide potential candidates for copying mechanism.

#### *3.2.1. Graph convolutional network*

In order to convert each document into a document-level graph, we use the following types of edges, as shown with different colors in Fig. 1.

- **Intra-sentence edge:** We apply dependency parser such as HanLP[1] to create dependency edges for each input sentence.
- **Inter-sentence edge:** We first connect the syntactic root of a sentence with the roots of the previous and next sentences. Second, we create edges between the same nouns in different sentences, since keyphrases are centered around nouns and we need to pay more attention to the nouns.
- **Self-looping edge:** Following the idea of self-looping in Kipf and Welling [11], we add edges that point to itself.

After the document-level graph is constructed, we attain an adjacency matrix **A** according to the words in the document. It is worth noting that intra-sentence edges are directed. To consider both non-direction-aware and direction-aware scenario,

[1] https://github.com/hankcs/HanLP

we propose two variants of GKG, i.e., GKG-DG on graphs which are non-direction, and GKG-DT concerning the direction-aware scenario.

For GKG-DG, we update the representation of $i$th node by applying GCN with normalization factor as below:

$$\boldsymbol{h}_i^l = ReLU(\sum_{j=1}^{n} \boldsymbol{A}_{ij}\boldsymbol{W}^l\boldsymbol{g}_j^{l-1}/(d_i+1)+\boldsymbol{b}^l) \tag{1}$$

where $\boldsymbol{g}_j^{l-1}$ denotes the $j$th node's representation evolved from the preceding GCN layer while $\boldsymbol{h}_i^l$ is the product of current layer, and $d_i$ is degree of the $i$th node. The initial representation for $\boldsymbol{h}_i^0$ is $\boldsymbol{x}_i$. $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$ are trainable parameters.

Practically, the difference between GKG-DG and GKG-DT lies in their adjacency matrices: GKG-DT needs to consider both the outgoing and incoming cases of nodes. We construct matrices of incoming and outgoing respectively, and obtain the hidden features $\overrightarrow{\boldsymbol{h}}_i^l$ and $\overleftarrow{\boldsymbol{h}}_i^l$ by using the two matrices and Eq. (1). Then, we combine both outgoing and incoming hidden features as the final hidden feature:

$$\boldsymbol{h}_i^l = \sigma(\boldsymbol{W}_h[\overrightarrow{\boldsymbol{h}}_i^l; \overleftarrow{\boldsymbol{h}}_i^l]+\boldsymbol{b}_h) \tag{2}$$

where $\boldsymbol{W}_h$ and $\boldsymbol{b}_h$ are trainable parameters, $\sigma$ is a nonlinear function.

It is worth mentioning that we do not directly use $\boldsymbol{h}_i^l$ as the input of successive GCN layer, but first perform a content-aware transformation:

$$\boldsymbol{g}_i^l = \mathcal{F}(\boldsymbol{h}_i^l) \tag{3}$$

where $\mathcal{F}(\cdot)$ is a function assigning attention weights for controlling information from last layer. The intuition is that if a node contains more useful information, the more information is propagated to other nodes. Specifically, the definition of function $\mathcal{F}(\cdot)$ is:

$$\alpha_i = (\boldsymbol{q}^\top\sigma(\boldsymbol{W}_1\boldsymbol{c}+\boldsymbol{W}_2\boldsymbol{h}_i^l+\boldsymbol{b})) \tag{4}$$

$$\mathcal{F}(\boldsymbol{h}_i^l) = \alpha_i\boldsymbol{h}_i^l \tag{5}$$

where $\alpha_i$ represents the degree of semantic importance between the node and document. The final output of the $L$-layer GCN is $\boldsymbol{H}^g = \{\boldsymbol{h}_1^L, \boldsymbol{h}_2^L, \dots, \boldsymbol{h}_n^L\}$. Finally, we use a readout function to aggregate a graph-level representation for the document, and its definition as below:

$$\boldsymbol{h}_g = Meanpooling(\boldsymbol{H}^g) + Maxpooling(\boldsymbol{H}^g) \tag{6}$$

We assume that each word plays a role in the document, and the core word should make more prominent contributions.

### 3.2.2. Candidate predictor network

Copying mechanism has been used in keyphrase generation, allowing model directly to extract words from the source input. Different from existing methods, we take a more aggressive approach to provide the candidate words for copying mechanism via candidate predictor network. That is, when preparing the training dataset, we need explicitly label a word in a target phrase as a candidate if it appears in both source input and target phrase. The intuition behind such an aggressive copying mechanism can be understood as we only need to copy the words that may form a keyphrase without considering the probability of unimportant words being copied.

Based on the nodes representations output from GCN, an unnormalized probability computed by a Multi-Layer-Perceptron (MLP) is used to predict which words should be preserved. Subsequently, a Straight-Through (ST) Gumbel-Softmax estimator is used to enforce a discrete selection, resulting in an $N$-dimensional binary vector, where 1 indicates that the word is a candidate.

Specifically, we first compute the probability of each word being a candidate, and then replace the argmax function with the differentiable softmax function.

$$\pi_{ij} = MLP(\boldsymbol{h}_i^L) \tag{7}$$

$$p_{ij} = \frac{exp(log(\pi_{ij}+g_j)/\tau)}{\sum_{t=1}^{k} exp(log(\pi_{it}+g_t)/\tau)} \tag{8}$$

$$g_j = -log(-log(u_j)) \tag{9}$$

$$u_j \sim Uniform(0,1) \tag{10}$$

where $g_j$ is randomly uniform Gumbel noise adding to the unnormalized scores, $\tau$ is a temperature parameter. When $\tau$ approaches zero, the Gumbel-Softmax distribution becomes a one-hot vector.

In the forward pass, the ST Gumbel-distribution is used to sample a one-hot vector by:

$$p_{ij}^{ST} = \begin{cases} 1, & j = argmax_t p_{it}; \\ 0, & otherwise. \end{cases} \tag{11}$$

Using ST Gumbel-Softmax, candidate predictor network can generate a $N$-dimensional binary vector $\boldsymbol{\beta}$ for each input document. Then the vector is fed into the sequence decoder as a candidate masking for the sequence of words in copying mechanism.

## 3.3. Sequence decoder

We adopt an unidirectional GRU (Uni-GRU) model with copying mechanism [7] as decoder.

### 3.3.1. Generating from vocabulary

The decoder takes the representation $\boldsymbol{H}^s$ from the sequence encoder as the attention memory and previously decoded words to generate the output sequence one word at a time. To make the decoder aware of input, we use document's representation to initialize the decoder hidden states.

$$\boldsymbol{s}_0 = \boldsymbol{c} \tag{12}$$

At each decoding step $j$, the model takes the embedding of the previous word $u_{j-1}$ and its previous hidden state $s_{j-1}$ as input and calculates current hidden state.

$$\boldsymbol{s}_j = GRU(\boldsymbol{u}_{j-1}, \boldsymbol{s}_{j-1}) \tag{13}$$

Next, computing attention weight $\alpha_{ij}$ for each word in the document,

$$e_{ij} = \boldsymbol{v}_\alpha^T tanh(\boldsymbol{W}_\alpha[\boldsymbol{h}_i^s; \boldsymbol{s}_j; \boldsymbol{h}_g]+\boldsymbol{b}_\alpha) \tag{14}$$

$$\boldsymbol{\alpha}_j = softmax(\boldsymbol{e}_j) \tag{15}$$

Here, $\boldsymbol{v}_\alpha$, $\boldsymbol{W}_\alpha$ and $\boldsymbol{b}_\alpha$ are trainable parameters. The attention weights indicate how the model spreads out the amount it cares about different encoder hidden states during decoding.

Then, we re-weight the word attention scores by incorporating the candidate predictor network results, and compute context vector:

$$\boldsymbol{\alpha}_j = \boldsymbol{\alpha}_j\boldsymbol{\beta} \tag{16}$$

$$\boldsymbol{c}_j = \boldsymbol{\alpha}_j^\top\boldsymbol{H}^s \tag{17}$$

From here, the decoder generates a word from a vocabulary distribution $P_{vocab}$ according to:

$$P_{vocab} = softmax(\boldsymbol{W}_{gen}[\boldsymbol{s}_j; \boldsymbol{c}_j; \boldsymbol{h}_g]+\boldsymbol{b}_{gen}) \tag{18}$$

### 3.3.2. Variational selector network

The above module generates target keyphrase words from a vocabulary. Another important way to generate target words is copying from input document. To allow copying the words form the source document, Gu et al. [7] introduce a copying mechanism into Seq2Seq model, allowing the model to use a soft switch to choose between generating from the vocabulary and copying from the input document [10]. In this scenario, the switch of the source of words can be viewed as a binary classification problem solved by selector. In previous work, the selector in the models is a deterministic mapping function, whereas we allow it to be variational. We assume that selector might optionally depend on an independent noisy input $\eta$, with a known fixed distribution like $\mathcal{N}(0,1)$.

Under the above conditions, we now adopt the idea of Variational Autoencoder (VAE) [13] to design our Variational Selector Network (VSN). We can interpret our VSN as part of VAE: an encoder first compresses the representation $\boldsymbol{v}_j = [\boldsymbol{c}_j; \boldsymbol{y}_{j-1}; \boldsymbol{s}_j; \boldsymbol{h}_g]$ into a continuous hidden vector $\boldsymbol{z}$, and then a selector outputs $\lambda_j$ as the probability of copying. Concretely, first, producing mean $\mu$ and variance $\sigma$ by an MLP network, which takes the context vector c together with word embedding y, decoder state s and graph representation h as input. Then, sampling latent variable $\boldsymbol{z}$ by the parameterized method from a multivariate Gaussian distribution. Eqs. (19) and (20) are the calculation formulas for $\boldsymbol{z}$:

$$\boldsymbol{z}_j = \boldsymbol{\mu} + \boldsymbol{\epsilon} \cdot \boldsymbol{\sigma}^2 \tag{19}$$

$$\boldsymbol{\mu} = f_\mu f_v(\boldsymbol{v}_j),\ log\boldsymbol{\sigma} = f_\sigma f_v(\boldsymbol{v}_j) \tag{20}$$

where $f_*$ is a neural perceptron with ReLU activated function, and $\boldsymbol{\epsilon} \in \mathcal{N}(\mathbf{0}, \mathbf{1})$.

Next, the probability of copying words from the document is given by:

$$\lambda_j = sigmoid(\boldsymbol{W}_\lambda \boldsymbol{z}_j + \boldsymbol{b}_\lambda) \tag{21}$$

where $\boldsymbol{W}_\lambda$ and $\boldsymbol{b}_\lambda$ being learning parameters.

Finally, we obtain distribution for predicting the $j$th target word with the formula as follow:

$$P_{final} = (1-\lambda_j) \cdot P_{vocab} + \lambda_j \cdot P_{copy} \tag{22}$$

where $P_{copy} = \boldsymbol{\alpha}_j$.

## 3.4. Training

Our model is trained by the standard gradient descent algorithm, and the loss function consists of four parts.

### 3.4.1. Maximum likelihood learning

Since the language generation task in our experiments is to produce corresponding keyphrases of the documents, which is a multi-label classification problem, we adopt the widely used negative log likelihood loss function as the core objective, denotes as $\mathcal{L}_{KG}$,

$$\mathcal{L}_{KG} = -\sum_{i=1}^{N}\sum_{j=1}^{L_y} log(P_{final}(y_{i,j}|y_{i,<j}, x_i; \theta)) \tag{23}$$

where $N$ is the size of training dataset, $L_y$ is the length of keyphrase, and $\theta$ denotes the learnable parameters of the model. In particular, we apply the teacher forcing strategy: at training time, the decoder inputs are previous tokens from the ground-truth; at test time, the inputs are previous tokens predicted by the decoder.

For candidate prediction task, as each prediction is a binary classification problem, the loss function for the $N$ training instances is defined by:

$$\mathcal{L}_{CP} = -\sum_{i=1}^{N}\sum_{j=1}^{L_x}[y_{ij}log(\hat{y}_{ij}) + (1-y_{ij})log(1-\hat{y}_{ij})] \tag{24}$$

where $y$ is the ground-truth label, $\hat{y}$ is the output of a softmax function.

### 3.4.2. Variational inference learning

For approximate inference of the $\boldsymbol{z}$ and $\boldsymbol{v}$ relationship, we use the variational selector network, which is a simplification of the VAE model. The VAE model leverages the variational inference to derive a negative variational lower bound [48,49] as objective function, and the whole loss consists of Kullback–Leibler divergence [50] loss and reconstruction loss. Since GKG does not need to reconstruct the original input v, we omit the reconstruction loss term and define the loss function for variational selector as follows:

$$\mathcal{L}_{KL} = D_{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{v})) \tag{25}$$

where $q(\boldsymbol{z})$ denotes a standard Gaussian distribution.

### 3.4.3. Maximizing mutual information learning

The learning objective is to maximize the MI between the document and its target keyphrase. We introduce an enhancement mechanism, Discriminator in Fig. 1, that contrasts document representation $\boldsymbol{c}$ with its target phrase representation $\boldsymbol{k}$ or another phrase representation $\boldsymbol{k}'$, and scores the consistency between them. Note that we use the last state $\boldsymbol{k}$ of decoder as keyphrase representation. By this contrast learning manner, GKG improves the association between document and its corresponding keyphrase, and reduces the association between document and other keyphrases. However, it is intractable to calculate MI in a high-dimensional space. In practice, there are some lower bound estimators [16,51,52] of MI to approximate MI. Following [16], we use the Jensen–Shannon estimator to maximize a lower bound on MI, and the formulation of Jensen–Shannon estimator as below:

$$\hat{\mathcal{I}}_\omega^{JSD}(\boldsymbol{k}; \boldsymbol{c}) := E_{\mathbb{P}}[-sp(-T_\omega(\boldsymbol{k}, \boldsymbol{c}))] - E_{\mathbb{P}\times\tilde{\mathbb{P}}}[sp(-T_\omega(\boldsymbol{k}', \boldsymbol{c}))] \tag{26}$$

where $T_\omega$ is a neural network with learnable parameters $\omega$. $k$ is a positive sample, $k'$ is a negative sample drawn from $\tilde{\mathbb{P}} = \mathbb{P}$, and $sp(z) = log(1+e^z)$ is the softplus function. In practice, we generate negative samples using the last states from other decoders in a batch. We define our MI learning objective on (document, keyphrase) pairs, and maximize the average estimated MI as follow:

$$\mathcal{L}_{MI} = argmax_\omega \frac{1}{N}(\sum_{i=1}^{N} \hat{\mathcal{I}}_\omega^{JSD}(\boldsymbol{k}; \boldsymbol{c})) \tag{27}$$

By combining Eqs. (23), (24), (25), and (27), the overall loss function of GKG is finally formulated as:

$$\mathcal{L} = \mathcal{L}_{KG} + \gamma_{CP} \cdot \mathcal{L}_{CP} + \gamma_{MI} \cdot \mathcal{L}_{MI} + \gamma_{KL} \cdot \mathcal{L}_{KL} \tag{28}$$

where the hyper-parameter $\gamma_{CP}$, $\gamma_{MI}$ and $\gamma_{KL}$ balances the effects of different parts of model. In all experiments, we always set the $\gamma_{CP}$, $\gamma_{MI}$ and $\gamma_{KL}$ as 0.001, 0.05 and 1. For inference, we adopt beam search and generate a ranking list of output keyphrases following Meng et al. [6].

**Table 1**
Datasets statistics. Train, Valid, and Test denotes training, development, and test set, respectively.

| Dataset | Total | Train | Valid | Test |
|---|---|---|---|---|
| Twitter | 44,113 | 35,290 | 4411 | 4412 |
| Weibo | 46,296 | 37,036 | 4630 | 4630 |
| StackExchange | 49,447 | 39,557 | 4945 | 4945 |

## 4. Experiments

### 4.1. Datasets and metrics

To comprehensively evaluate the effectiveness of our model, we utilize three real-world datasets collected from different social media. Twitter and Weibo are microblogs for opinions on public affairs or users' daily lives, while StackExchange is one of the most popular Q&A sites for asking questions and obtaining answers from other users. For our experiments, we follow the training, validation and test split used in Wang et al. [10]. The detailed statistics are summarized in Table 1.

Compared with previous works, we employ macro-average F1@K and mean average precision (MAP) as automated evaluation metrics. F1@K compares the top K predicted keyphrases with the ground-truth keyphrases. Following Wang et al. [10], we report F1@1 and F1@3 scores for Twitter and Weibo, and F1@3 and F1@5 scores for StackExchange. MAP is measured over the top 5 predictions for all three datasets. Before the comparison, we apply the Porter Stemmer algorithm [6] to stem both predicted and ground truth keyphrases to a base form. A predicted keyphrase is considered correct or incorrect, depending on whether there is an exact matching keyphrase in the same position in the ground-truth.

### 4.2. Baselines and state of the art

To comprehensively evaluate our model, we compare the two variants of our model with the following baseline methods and state-of-the-art models. Note that the results from some models are directly taken from [10].

- **Majority** [10]: this is a baseline which selects the top K keyphrases as test samples' keyphrases according to the frequency of key phrases in the training data.
- **TextRank** [24]: the network is constructed by the adjacent relations between words, and then the keyphrases are obtained by the PageRank algorithm.
- **KEA system** [53]: identification of candidate keyphrases based on lexical features.
- **TF-IDF** [10]: a method that ranks n-grams (n = 1,2,3) in the source document statistically.
- **SEQ-TAG** [27]: a neural state-of-the-art sequence tagging keyphrase extraction model.
- **Seq2Seq-Copy** [6]: an RNN-based encoder–decoder model with a copying mechanism.
- **Seq2Seq-Corr** [9]: an extension of Seq2Seq-Copy which incorporates review and coverage mechanism to capture correlation among multiple keyphrases.
- **TG-NET** [54]: another extension of Seq2Seq-Copy which employs the title to guide the generation process.
- **TAKG** (Topic-Aware Neural Keyphrase Generation) [10]: a hybrid system that infuses topic information into the encoder–decoder framework.
- **Transformer** [55]: a transformer architecture with the same configuration as encoder and decoder.

### 4.3. Preprocessing and experiment setup

We implement GKG in PyTorch [56]. We use the source codes published by the authors, and try to tune them to their best performance. Each document of the dataset is preprocessed with HanLP to obtain POS and dependency tree. We perform tokenization and define a vocabulary following Wang et al. [10], where 30K most frequent words for Twitter, and 50K for Weibo and StackExchange each.

In all experiments, we use Adam [57] as the optimizer with initial learning rate 1e−3. Gradient clipping 1.0 and gradient norm tokens are utilized during training. 10 training epochs were performed with early-stopping strategy [58], and batch size set to 64. The number of RNN encoding layers, graph encoding layers and RNN decoding layers are set as 2, 3, and 1, respectively. The number of hidden units in Uni-GRU is set as 300 and each direction is 150 for Bi-GRU. We use pretrained word embeddings from GloVe,[2] and POS embeddings is randomly initialized which size is set to 50 for training. We empirically set the $\gamma_{CP} = 0.001$, $\gamma_{MI} = 0.05$, $\gamma_{KL} = 1$ for balancing the parts of the loss. Here, we trained our models on a single NVIDIA Tesla GPU. For three datasets, the validation loss reaches its minimum between 3 to 4 epochs, each training epoch of the Twitter dataset takes around 50 s, the Weibo dataset takes about 90 s, and the StackExchange dataset takes approximately 190 s.

For transformer [55], our implementation is based on OpenNMT [59]. The Adam optimizer with beta1 = 0.9 and beta2 = 0.999 is used to update parameters. The learning rate and warmup steps are set as 0.1 and 4000. After the warm-up stage the inverse square root learning rate scheduler is applied. The number of hidden units is set as 256, the feed-forward hidden size is 1024, and the number of heads is 8. The number of encoding layers and decoding layers are set as 6 and 6, respectively. Other model configurations are in line with models RNN-based. For training, we stop the training process when the validation loss does not reduce. During testing, we use beam search with beam size 50 and length penalty with factor 0.6.

## 5. Results

In this section, we first compare our proposed models with other methods in terms of different evaluation metrics. Then, we evaluate our models' performance on present and absent keyphrase predictions. We also analyze the effects of the number of GCN layers, pooling functions, candidate words and intra-sentence dependency on the model. Additionally, we analyze the keyphrases generated by GKG-DG and the baselines. Finally, we discuss the level of contribution that each model component imparts to the performance.

### 5.1. Comparison with baseline models

Table 2 presents our primary experimental results, which are compared with the baseline models. From the results, we can observe that:

We note that the performance of GKG-DG surpasses that of the state-of-the-art KG models on the Twitter, Weibo and StackExchange datasets. The results in the table demonstrate that our proposed graph-based KG model, which incorporates the dependency structure information in the document, can be effectively utilized to improve performance.

More specifically, when compared with the Weibo dataset, we achieve greater improvements for the Twitter and StackExchange datasets. For example, for the Twitter dataset, the F1@1 score is

[2] https://nlp.stanford.edu/projects/glove/.

**Table 2**
Performance comparison of different models. Average result scores (%) displayed over 5 runs with random initialization. The subscript represents the corresponding standard deviation (e.g., $40.61_{0.6}$ means 40.61 ± 0.6). The description for the models is given in Section 4.2. Boldface scores in each column indicate the best results.

| Models | Twitter | | | Weibo | | | StackExchange | | |
|---|---|---|---|---|---|---|---|---|---|
| | F1@1 | F1@3 | MAP | F1@1 | F1@3 | MAP | F1@3 | F1@5 | MAP |
| **Baselines** | | | | | | | | | |
| Majority [10] | 9.36 | 11.85 | 15.22 | 4.16 | 3.31 | 5.47 | 1.79 | 1.89 | 1.59 |
| TextRank [24] | 1.16 | 1.14 | 1.89 | 1.90 | 1.51 | 2.46 | 13.50 | 12.74 | 12.61 |
| KEA system [53] | 1.73 | 1.94 | 1.89 | 0.18 | 0.49 | 0.57 | 6.03 | 8.28 | 4.76 |
| TF-IDF [10] | 0.50 | 0.56 | 0.50 | 0.20 | 0.20 | 0.20 | 15.80 | 15.23 | 14.25 |
| **State of the arts** | | | | | | | | | |
| SEQ-TAG [27] | $22.79_{0.3}$ | $12.27_{0.2}$ | $22.44_{0.3}$ | $16.34_{0.2}$ | $8.99_{0.1}$ | $16.53_{0.3}$ | $17.58_{1.6}$ | $12.82_{1.2}$ | $19.03_{1.3}$ |
| Seq2Seq [6] | $34.10_{0.5}$ | $26.01_{0.3}$ | $41.11_{0.3}$ | $28.17_{1.7}$ | $20.59_{0.9}$ | $34.19_{1.7}$ | $22.99_{0.3}$ | $20.65_{0.2}$ | $23.95_{0.3}$ |
| Seq2Seq-Copy [6] | $36.60_{1.1}$ | $26.79_{0.5}$ | $43.12_{1.2}$ | $32.01_{0.3}$ | $22.69_{0.2}$ | $38.01_{0.1}$ | $31.53_{0.1}$ | $27.41_{0.2}$ | $33.45_{0.1}$ |
| Seq2Seq-Corr [9] | $34.97_{0.8}$ | $26.13_{0.4}$ | $41.64_{0.5}$ | $31.64_{0.7}$ | $22.24_{0.5}$ | $37.47_{0.8}$ | $30.89_{0.3}$ | $26.97_{0.2}$ | $32.87_{0.6}$ |
| TG-NET [54] | – | – | – | – | – | – | $32.02_{0.3}$ | $27.84_{0.3}$ | $34.05_{0.4}$ |
| TAKG [10] | $38.49_{0.3}$ | $27.84_{0.0}$ | $45.12_{0.2}$ | $34.99_{0.3}$ | $24.42_{0.2}$ | $41.29_{0.4}$ | $33.41_{0.2}$ | $29.16_{0.1}$ | $35.52_{0.1}$ |
| Transformer | $32.67_{1.1}$ | $25.26_{1.1}$ | $38.63_{1.1}$ | $26.86_{0.6}$ | $19.30_{0.7}$ | $31.56_{0.6}$ | $24.96_{0.6}$ | $21.41_{0.4}$ | $25.84_{0.6}$ |
| **Ours** | | | | | | | | | |
| GKG-DT | $37.90_{0.0}$ | $29.41_{0.2}$ | $46.31_{0.2}$ | $35.30_{0.1}$ | $24.90_{0.3}$ | $41.96_{0.2}$ | $\mathbf{35.33_{0.2}}$ | $\mathbf{31.16_{0.1}}$ | $\mathbf{37.96_{0.2}}$ |
| GKG-DG | $\mathbf{40.61_{0.6}}$ | $\mathbf{29.77_{0.3}}$ | $\mathbf{48.04_{0.7}}$ | $\mathbf{35.59_{0.4}}$ | $\mathbf{25.06_{0.2}}$ | $\mathbf{42.25_{0.2}}$ | $34.75_{0.2}$ | $30.89_{0.2}$ | $37.06_{0.3}$ |

40.61, the F1@3 score is 29.77, and the MAP is 48.04; these values increase by 2.12, 1.93 and 2.92, respectively (relative to TAKG), but on Weibo, the score increase by 0.6, 1.64 and 0.96, respectively. One possible reason is that the provided Weibo dataset has removed punctuation, which breaks the syntactic dependence in the sentence. However, the construction method of the graph proposed in this paper relies on the syntactic dependence, and it is not completely applicable to the documents in the Weibo dataset. In contrast, the syntactic structures in the Twitter and StackExchange datasets are more complete; thus, GKG can obtain better dependency information. We conjecture that this is likely the reason GKG-DG obtains better results on the Twitter and StackExchange datasets.

Meanwhile, compared with GKG-DT, GKG-DG performs better by a large margin on the Weibo and Twitter datasets. We find that the F1@1 score of GKG-DT on the Twitter dataset is only 37.90, which is even lower than that of TAKG. We suspect that, for text structure data, the information from the parent node to the child node is equally important as from child the node to the parent node; thus, only the one-way transmission of information causes a loss of information.

Interestingly, GKG-DT performs better than GKG-DG on the StackExchange dataset. From Table 2, we notice that the values of F1@3, F1@5, and MAP from GKG-DT are 35.33, 31.16, 37.96, respectively, which are slightly higher than the 34.75, 30.89 and 37.06 from GKG-DG. We suspect that, when compared with a short text, a long text causes more attention to be paid to the direction of information transmission, and the contribution of information from the parent and child nodes to the final representation is different.

### 5.2. Present and absent keyphrases prediction

Additionally, to further validate the effectiveness of our model, we also evaluate the performance of present and absent keyphrase predictions separately (the results are shown in Figs. 2 and 3, respectively). On all datasets, our model demonstrates significant improvements in predicting absent and present keyphrases. For the Twitter, Weibo and StackExchange datasets, the F1@1 scores of GKG-DG are 87.34, 84.18, and 55.35. Relatively, the F1@1 scores of TAKG/Seq2Seq-Copy are 84.93/84.99, 81.44/83.27 and 53.35/54.09. We think that a possible reason for

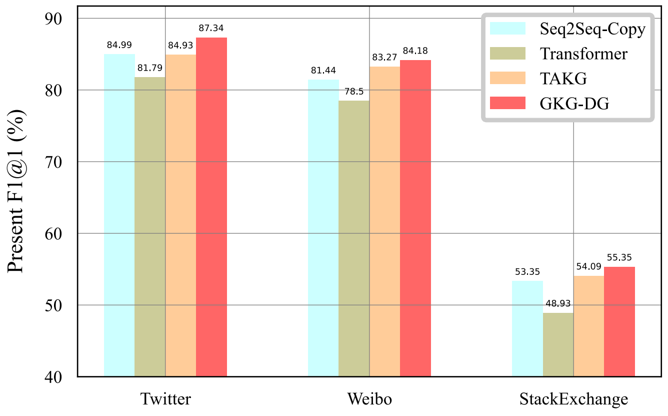

**Fig. 2.** Present keyphrases prediction results on all datasets with F1@1 (%).

this phenomenon is that the probability of the copying mechanism copying wrong words is reduced by using candidates to mask irrelevant words in a document.

To our surprise, the improvement of metric R@5 is more observable. In particular, the R@5 on the Twitter dataset reaches up to 48.55, which shows that the modules proposed in this work can help our model make better decisions about whether to copy or not.

### 5.3. Analysis of GCN layers

Since GKG contains an L-layer GCN, we conduct an experiment to study how the number of GCN layers influences the performance of GKG-DG. In this comparison, we vary the value of L in the set {1,2,3,4,6,8,12} and check the corresponding F1@1 on the Twitter and Weibo datasets and F1@3 on StackExchange dataset. Fig. 4 presents the results of using different numbers of GCN layers. According to the line chart, it is clear that GKG-DG achieves the best performance when L is 3, which proves that the selection of the layer number in the experiment is reasonable. Using the multi-layer convolution process in GKG-DG, we can aggregate the information in more hops and therefore, achieve better performance. Since it is clear that the model's performance improves as the number of GCN layers increases, it is inadequate

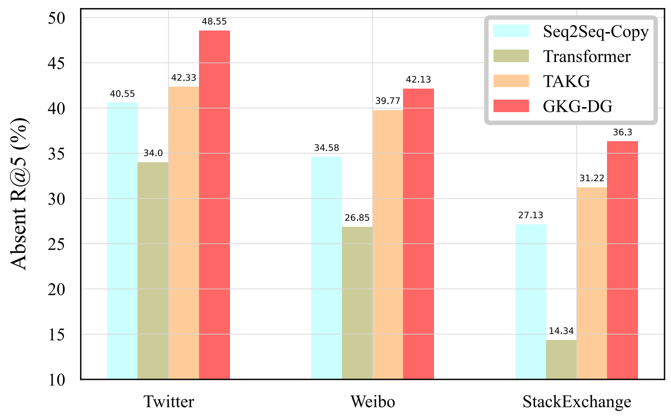

**Fig. 3.** Absent keyphrases prediction results on all datasets with R@5 (%).

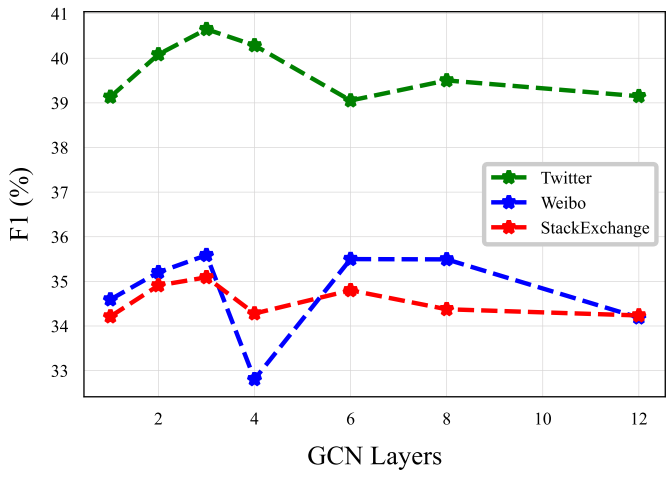

**Fig. 4.** Effect of the number of GCN layers.

to simply incorporate the information from its first-order and second-order neighboring words to generate the representation for each node. However, we observe that the performance becomes unstable and exhibits a decreasing trend when the number of GCN layers is larger than 3. When the number of GCN layers increases, the representations of neighboring nodes in the document-level graph will become increasingly similar since they are all calculated using those of their neighbors in the document-level graph. The problem of over-smoothing emerges when the number of layers rises beyond a threshold, explaining the F1 drop after addition of more than 3 layers, as shown in the figure.

### 5.4. Avg-Max pooling vs. Average pooling vs. Max pooling

In Section 3.2.1, we apply average pooling [60] and max pooling [61] to the outputs (last-layer hidden states) from GCN to generate graph representation for the input document. The results in Table 3 indicate the scores of different pooling functions when they are applied to GKG-DG on the Twitter dataset. Comparing Average (average pooling) and Max (max pooling), we notice that the performance of max pooling is significantly better than that of average pooling. We attribute this phenomenon to the fact that average pooling takes all words into account, which results in an average value that may not be suitable for documents, while max pooling focuses on the most important features. On the other hand, only relying on core words is not optimal, because we also observe that Avg-Max (combining average pooling and max pooling) performs slightly better than max

**Table 3**
Results of Twitter dataset with different pooling functions on GKG-DG. R@5-A: R@5 for absent keyphrases. The best scores (%) are bold.

| Pooling | F1@1 | F1@3 | R@5-A |
|---|---|---|---|
| Average | 38.46 | 28.34 | 45.74 |
| Max | 40.40 | 29.48 | 46.64 |
| Avg-Max | **40.61** | **29.77** | **48.55** |

**Table 4**
Keyphrase prediction results on Twitter. The best scores (%) are bold.

| Models | F1@1 | F1@3 | R@5-A |
|---|---|---|---|
| GKG-DG-doc | 40.36 | 29.57 | 47.49 |
| GKG-DG | **40.61** | **29.77** | **48.55** |

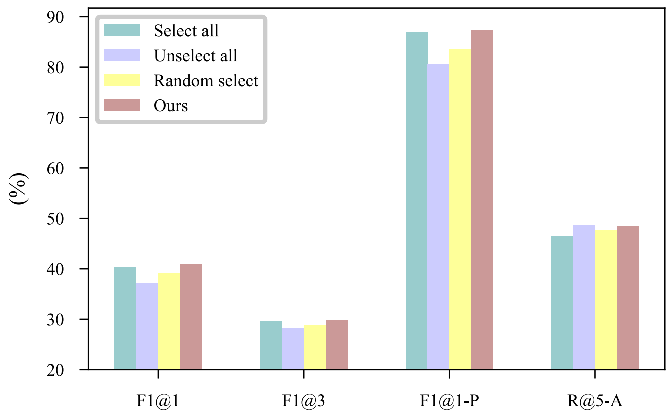

**Fig. 5.** Results of GKG-DG using different candidate words on Twitter dataset. F1@1-P: F1@1 for present keyphrases; R@5-A: R@5 for absent keyphrases.

pooling. Thus, we can conclude that every word plays a role in the document but that the core words contributes more explicitly.

### 5.5. Importance analysis of candidate word selection

To investigate whether GKG can distinguish the importance of candidate words, we compare the experimental results of different candidate selections, as shown in Fig. 5. We find that the model that uses predictor results as candidate words achieves the highest F1@1 and R@5 scores. At the same time, we observe that the performance of randomly selected candidate words is lower than that which selects all words. The results are consistent with our intuition that if we do not filter out words, the model is provided with the full context. However, we use the predictor to selectively retain words and filter out noisy words, which contributes to the increase in the F1 score.

### 5.6. Effects of intra-sentence dependence analysis

To confirm that our construction of intra-sentence dependency edges is optimal, we investigate the result of the different inputs used in the dependency parser. For GKG-DG-doc, we use the document as the input for the parser and do not distinguish between intra-sentence edges and inter-sentence edges. Table 4 summarizes the comparison results on the Twitter dataset, which show that GKG-DG achieves better performance than GKG-DG-doc and that GKG-DG can surpass GKG-DG-doc by approximately 0.3 on F1@1. These results demonstrate that document-level graphs with richer relations are more helpful for obtaining dependency information.

**Table 5**
An example of generated keyphrases by baselines and GKG-DG. The correct prediction is underlined. Only the first five keyphrases of the prediction are shown in this table.

| Source text | this is crap. i wana see more cameramen get knocked out. |
|---|---|
| Target | super bowl |
| Seq2Seq | fail; random; now playing; quote alicious; egypt; |
| Seq2Seq-Copy | fail; just saying; take me out; <u>super bowl</u>; eastenders; |
| Transformer | sotu; fail; tcot; egypt; random; |
| TAKG | big fat gypsy weddings; tcap; rhoa; nt as; aus open; |
| **GKG-DG** | <u>super bowl</u>; fail; random; just saying; good night; |

### 5.7. Case study

For a more comprehensive comparison, in Table 5, we show a case text as well as some keyphrases generated by the different models. We notice that the ground truth phrase "*super bowl*" does not appear in the source text. As observed, only our GKG-DG and Seq2Seq-Copy correctly generate "*super bowl*" as a keyphrase, while the other three baselines output false predictions. In addition, we observe that "*super bowl*" is ranked first in the GKG-DG prediction but not in Seq2Seq-Copy predict. Our GKG-DG generates more accurate keyphrases for the text, which demonstrates that our proposed model is more powerful in predicting absent keyphrases.

### 5.8. Ablation study

To further examine the level of contribution that each model component impacts to the performance, an ablation study is performed with GKG-DG on the Twitter dataset. Table 6 reports the performance of the full GKG-DG model and its ablations. We also present the results of Seq2Seq with copying mechanism as a baseline. To ablate the variational selector network, we replace it with the deterministic network (i.e., GKG-DG $w/o$ $vas$), which effectively leads to a Seq2Seq model with the copying mechanism from the documents (pointer-generator). This ablation results in a significant drop in F1@1 to 38.92, F1@1-P to 87.03 and R@5-A to 47.66, confirming the superiority of the variational selector network over the vanilla pointer-generator in leveraging the document source to generate keyphrase. In addition, we ablate the graph encoder and compare the results of GKG-DG $w/o$ $gcn$ (i.e., not using the graph encoder) with those from the full model; this comparison results in a drop of approximately 0.7 and 1.1 on F1@1-A and R@5-A when the graph encoder is not used. We also observe that GKG-DG $w/o$ $gcn$ achieves the highest score on F1@1-P, indicating that dependency information can help the model to better balance "copy" and "generate" words to improve the overall performance. From the ablation results, it can be concluded that the graph encoder module contributes to GKG to a considerable extent since GCN captures dependency information.

To understand more directly the impact of the VSN, we train both original Seq2Seq-Copy and Seq2Seq-VSN (replacing deterministic network with variational selector network) models. For this experiment, we record the model checkpoints for every epoch during training and calculate the F1 score on all datasets. The experiments are repeated three times using different random seeds. The trajectory of the F1 value with epoch number increasing are plotted in Fig. 6. The $x$-axis is epoch number and $y$-axis is F1 score. We observe that as the epoch increases, the F1 score curve of Seq2Seq-VSN will eventually be higher than that of Seq2Seq-Copy trained using a deterministic network. This indicates the benefit of using variational mapping function to determine the final selection probability for each word in a phrase.

**Table 6**
Ablation using GKG-DG on Twitter. $w/o$ $gcn$: preserve variational selector network, but not use graph encoder; $w/o$ $vas$: only preserve graph encoder. F1@1-P: F1@1 for present keyphrases; R@5-A: R@5 for absent keyphrases. The best scores (%) are bold.

| Models | F1@1 | F1@1-P | R@5-A |
|---|---|---|---|
| Seq2Seq-Copy | 36.60 | 84.99 | 40.55 |
| GKG-DG $w/o$ $gcn$ | 39.97 | **88.07** | 47.47 |
| GKG-DG $w/o$ $vas$ | 38.92 | 87.03 | 47.66 |
| GKG-DG | **40.61** | 87.34 | **48.55** |

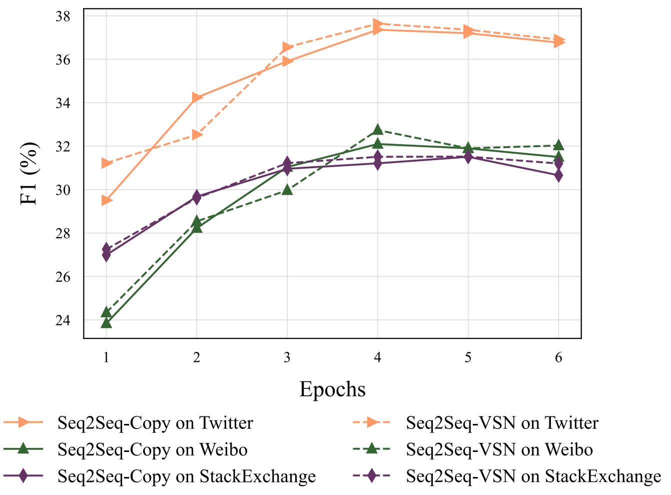

**Fig. 6.** Performances of Seq2Seq-Copy using different mapping function.

## 6. Conclusion and future work

We propose a novel Keyphrase Generation (KG) model based on GCN and mutual information maximization, which fuses dependency information captured by GCN with context information to alleviate the problem of social text information scatter, and maximizes the MI between the text and target keyphrase to ensure the consistency between them. Furthermore, we utilize variational selector network to determine the final selection probability of each word in keyphrase, which depends on its probabilities of copying from a given document and being generated from a vocabulary. Experimental results showed that GKG achieves better performance than the state-of-the-art methods on Twitter, Weibo and StackExchange datasets. As future work, we will design a specific graph neural network that takes the edge information between words account for better encoding representations of nodes.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was supported in part by the Consulting Project of Chinese Academy of Engineering under Grant 2020-XY-5, 2018-XY-07, and in part by the Fundamental Research Funds for the Central Universities, China, the Collaborative Innovation Center of Novel Software Technology and Industrialization, China.

## References

[1] D. Kang, E. Hovy, Plan ahead: Self-supervised text planning for paragraph completion task, in: Proceedings of EMNLP, Association for Computational Linguistics, Online, 2020, pp. 6533–6543, http://dx.doi.org/10.18653/v1/2020.emnlp-main.529.

[2] X. Meng, F. Wei, X. Liu, M. Zhou, S. Li, H. Wang, Entity-centric topic-oriented opinion summarization in twitter, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012, pp. 379–387.

[3] X. Hua, Z. Hu, L. Wang, Argument generation with retrieval, planning, and realization, in: Proceedings of ACL, Association for Computational Linguistics, Florence, Italy, 2019, pp. 2661–2672, http://dx.doi.org/10.18653/v1/P19-1255.

[4] B. Liu, M. Zhao, D. Niu, K. Lai, Y. He, H. Wei, Y. Xu, Learning to generate questions by learningwhat not to generate, in: The World Wide Web Conference, in: WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1106–1118, http://dx.doi.org/10.1145/3308558.3313737.

[5] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, in: NIPS'14, MIT Press, Cambridge, MA, USA, 2014, pp. 3104–3112.

[6] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, Y. Chi, Deep keyphrase generation, in: Proceedings of ACL (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 582–592, http://dx.doi.org/10.18653/v1/P17-1054.

[7] J. Gu, Z. Lu, H. Li, V.O. Li, Incorporating copying mechanism in sequence-to-sequence learning, in: Proceedings of ACL (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1631–1640, http://dx.doi.org/10.18653/v1/P16-1154.

[8] Y. Zhang, W. Xiao, Keyphrase generation based on deep seq2seq model, IEEE Access 6 (2018) 46047–46057.

[9] J. Chen, X. Zhang, Y. Wu, Z. Yan, Z. Li, Keyphrase generation with correlation constraints, in: Proceedings of EMNLP, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 4057–4066, http://dx.doi.org/10.18653/v1/D18-1439.

[10] Y. Wang, J. Li, H.P. Chan, I. King, M.R. Lyu, S. Shi, Topic-aware neural keyphrase generation for social media language, in: Proceedings of ACL, Association for Computational Linguistics, Florence, Italy, 2019, pp. 2516–2526, http://dx.doi.org/10.18653/v1/P19-1240.

[11] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, OpenReview.net, 2017.

[12] I. Ni'mah, V. Menkovski, M. Pechenizkiy, BSDAR: Beam search decoding with attention reward in neural keyphrase generation, 2019, arXiv preprint arXiv:1909.09485.

[13] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: International Conference on Learning Representations, 2014.

[14] K. Chawla, D. Yang, Semi-supervised formality style transfer using language model discriminator and mutual information maximization, in: Proceedings of EMNLP, Association for Computational Linguistics, Online, 2020, pp. 2340–2354, http://dx.doi.org/10.18653/v1/2020.findings-emnlp.212.

[15] P. velickovic, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, in: International Conference on Learning Representations, 2019.

[16] R.D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, Y. Bengio, Learning deep representations by mutual information estimation and maximization, in: International Conference on Learning Representations, 2019.

[17] F.-Y. Sun, J. Hoffman, V. Verma, J. Tang, InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization, in: International Conference on Learning Representations, 2020.

[18] K.S. Hasan, V. Ng, Automatic keyphrase extraction: A survey of the state of the art, in: Proceedings of ACL, (Volume 1: Long Papers), Association for Computational Linguistics, Baltimore, Maryland, 2014, pp. 1262–1273, http://dx.doi.org/10.3115/v1/P14-1119.

[19] K. Barker, N. Cornacchia, Using noun phrase heads to extract document keyphrases, in: Conference of the Canadian Society for Computational Studies of Intelligence, Springer, 2000, pp. 40–52.

[20] M. Grineva, M. Grinev, D. Lizorkin, Extracting key terms from noisy and multitheme documents, in: Proceedings of the 18th International Conference on World Wide Web, in: WWW '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 661–670, http://dx.doi.org/10.1145/1526709.1526798.

[21] O. Medelyan, E. Frank, I.H. Witten, Human-competitive tagging using automatic keyphrase extraction, in: Proceedings of EMNLP, Association for Computational Linguistics, Singapore, 2009, pp. 1318–1327.

[22] X. Jiang, Y. Hu, H. Li, A ranking approach to keyphrase extraction, in: Proceedings of ACM SIGIR, in: SIGIR '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 756–757, http://dx.doi.org/10.1145/1571941.1572113.

[23] J. Villmow, M. Wrzalik, D. Krechel, Automatic keyphrase extraction using recurrent neural networks, in: International Conference on Machine Learning and Data Mining in Pattern Recognition, Springer, 2018, pp. 210–217.

[24] R. Mihalcea, P. Tarau, TextRank: Bringing order into text, in: Proceedings of EMNLP, Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 404–411.

[25] Z. Liu, P. Li, Y. Zheng, M. Sun, Clustering to find exemplar terms for keyphrase extraction, in: Proceedings of EMNLP, Association for Computational Linguistics, Singapore, 2009, pp. 257–266.

[26] Z. Liu, W. Huang, Y. Zheng, M. Sun, Automatic keyphrase extraction via topic decomposition, in: Proceedings of EMNLP, Association for Computational Linguistics, Cambridge, MA, 2010, pp. 366–376.

[27] Q. Zhang, Y. Wang, Y. Gong, X. Huang, Keyphrase extraction using deep recurrent neural networks on Twitter, in: Proceedings of EMNLP, Association for Computational Linguistics, Austin, Texas, 2016, pp. 836–845, http://dx.doi.org/10.18653/v1/D16-1080.

[28] Y. Luan, M. Ostendorf, H. Hajishirzi, Scientific information extraction with semi-supervised neural tagging, in: Proceedings of EMNLP, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 2641–2651, http://dx.doi.org/10.18653/v1/D17-1279.

[29] H.P. Chan, W. Chen, L. Wang, I. King, Neural keyphrase generation via reinforcement learning with adaptive rewards, in: Proceedings of ACL, Association for Computational Linguistics, Florence, Italy, 2019, pp. 2163–2174, http://dx.doi.org/10.18653/v1/P19-1208.

[30] J. Zhao, Y. Zhang, Incorporating linguistic constraints into keyphrase generation, in: Proceedings of ACL, Association for Computational Linguistics, Florence, Italy, 2019, pp. 5224–5233, http://dx.doi.org/10.18653/v1/P19-1515.

[31] P. Battaglia, R. Pascanu, M. Lai, D.J. Rezende, K. kavukcuoglu, Interaction networks for learning about objects, relations and physics, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, in: NIPS'16, Curran Associates Inc., Red Hook, NY, USA, 2016, pp. 4509–4517.

[32] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, in: ICML'17, JMLR.org, 2017, pp. 1263–1272.

[33] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, in: NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 1025–1035.

[34] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, Stat 1050 (2018) 4.

[35] E. Alsentzer, S. Finlayson, M. Li, M. Zitnik, Subgraph neural networks, in: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 8017–8029.

[36] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, Q. Yang, Large-scale hierarchical text classification with recursively regularized deep graph-CNN, in: Proceedings of the 2018 World Wide Web Conference, in: WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, pp. 1063–1072, http://dx.doi.org/10.1145/3178876.3186005.

[37] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 7370–7377, http://dx.doi.org/10.1609/aaai.v33i01.33017370, (01).

[38] Y. Zhang, X. Yu, Z. Cui, S. Wu, Z. Wen, L. Wang, Every document owns its structure: Inductive text classification via graph neural networks, in: Proceedings of ACL, Association for Computational Linguistics, Online, 2020, pp. 334–339, http://dx.doi.org/10.18653/v1/2020.acl-main.31.

[39] C. Zhang, Q. Li, D. Song, Aspect-based sentiment classification with aspect-specific graph convolutional networks, in: Proceedings of EMNLP-IJCNLP, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 4568–4578, http://dx.doi.org/10.18653/v1/D19-1464.

[40] W. Li, J. Xu, Y. He, S. Yan, Y. Wu, X. Sun, Coherent comments generation for Chinese articles with a graph-to-sequence model, in: Proceedings of ACL, Association for Computational Linguistics, Florence, Italy, 2019, pp. 4843–4852, http://dx.doi.org/10.18653/v1/P19-1479.

[41] F. Christopoulou, M. Miwa, S. Ananiadou, Connecting the dots: Document-level neural relation extraction with edge-oriented graphs, in: Proceedings of EMNLP-IJCNLP, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 4925–4936, http://dx.doi.org/10.18653/v1/D19-1498.

[42] L. Qiu, Y. Xiao, Y. Qu, H. Zhou, L. Li, W. Zhang, Y. Yu, Dynamically fused graph network for multi-hop reasoning, in: Proceedings of ACL, Association for Computational Linguistics, Florence, Italy, 2019, pp. 6140–6150, http://dx.doi.org/10.18653/v1/P19-1617.

[43] R. Linsker, Self-organization in a perceptual network, Computer 21 (3) (1988) 105–117, http://dx.doi.org/10.1109/2.36.

[44] K. Hassani, A.H. Khasahmadi, Contrastive multi-view representation learning on graphs, in: International Conference on Machine Learning, PMLR, 2020, pp. 4116–4126.

[45] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, J. Huang, Graph representation learning via graphical mutual information maximization, in: Proceedings of the Web Conference 2020, 2020, pp. 259–270.

[46] D.B. Lee, S. Lee, W.T. Jeong, D. Kim, S.J. Hwang, Generating diverse and consistent QA pairs from contexts with information-maximizing hierarchical conditional VAEs, in: Proceedings of ACL, Association for Computational Linguistics, Online, 2020, pp. 208–224, http://dx.doi.org/10.18653/v1/2020.acl-main.20.

[47] Y. Zhang, R. He, Z. Liu, K.H. Lim, L. Bing, An unsupervised sentence embedding method by mutual information maximization, in: Proceedings of EMNLP, Association for Computational Linguistics, Online, 2020, pp. 1601–1610, http://dx.doi.org/10.18653/v1/2020.emnlp-main.124.

[48] D.M. Blei, A. Kucukelbir, J.D. McAuliffe, Variational inference: A review for statisticians, J. Amer. Statist. Assoc. 112 (518) (2017) 859–877.

[49] Y. Miao, E. Grefenstette, P. Blunsom, Discovering discrete latent topics with neural variational inference, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, in: ICML'17, JMLR.org, 2017, pp. 2410–2419.

[50] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.

[51] A.v.d. Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, 2018, arXiv preprint arXiv:1807.03748.

[52] M.I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, R.D. Hjelm, Mine: mutual information neural estimation, 2018, arXiv preprint arXiv:1801.04062.

[53] I.H. Witten, G.W. Paynter, E. Frank, C. Gutwin, C.G. Nevill-Manning, Kea: Practical automated keyphrase extraction, in: Design and Usability of Digital Libraries: Case Studies in the Asia Pacific, IGI global, 2005, pp. 129–152.

[54] W. Chen, H.P. Chan, P. Li, L. Bing, I. King, An integrated approach for keyphrase generation via exploring the power of retrieval and extraction, in: Proceedings of NAACL, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 2846–2856, http://dx.doi.org/10.18653/v1/N19-1292.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, u. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, in: NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 6000–6010.

[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., PyTorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019) 8026–8037.

[57] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations, 2015.

[58] R. Caruana, S. Lawrence, L. Giles, Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping, Adv. Neural Inf. Process. Syst. (2001) 402–408.

[59] G. Klein, Y. Kim, Y. Deng, J. Senellart, A. Rush, OpenNMT: Open-source toolkit for neural machine translation, in: Proceedings of ACL 2017, System Demonstrations, Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 67–72, URL https://aclanthology.org/P17-4012.

[60] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Handwritten digit recognition with a back-propagation network, in: Proceedings of the 2nd International Conference on Neural Information Processing Systems, in: NIPS'89, MIT Press, Cambridge, MA, USA, 1989, pp. 396–404.

[61] M.A. Ranzato, Y.-L. Boureau, Y. LeCun, Sparse feature learning for deep belief networks, in: Proceedings of the 20th International Conference on Neural Information Processing Systems, in: NIPS'07, Curran Associates Inc., Red Hook, NY, USA, 2007, pp. 1185–1192.