HTKG: Deep Keyphrase Generation with Neural Hierarchical Topic Guidance

Yuxiang Zhang
Civil Aviation University of China
Tianjin, China
yxzhang@cauc.edu.cn

Tao Jiang
Civil Aviation University of China
Tianjin, China
tjiang98@outlook.com

Tianyu Yang
Civil Aviation University of China
Tianjin, China
void612@126.com

Xiaoli Li
Institute for Infocomm Research/Centre for Frontier AI Research, A*STAR, Singapore
xlli@i2r.a-star.edu.sg

Suge Wang
Shanxi University
Taiyuan, China
wsg@sxu.edu.cn

ABSTRACT

Keyphrases can concisely describe the *high-level topics* discussed in a document that usually possesses hierarchical topic structures. Thus, it is crucial to understand the hierarchical topic structures and employ it to guide the keyphrase identification. However, integrating the *hierarchical topic information* into a deep keyphrase generation model is unexplored. In this paper, we focus on how to effectively exploit the hierarchical topic to improve the keyphrase generation performance (HTKG). Specifically, we propose a novel hierarchical topic-guided variational neural sequence generation method for keyphrase generation, which consists of two major modules: a *neural hierarchical topic model* that learns the latent topic tree across the whole corpus of documents, and a *variational neural keyphrase generation model* to generate keyphrases under hierarchical topic guidance. Finally, these two modules are jointly trained to help them learn complementary information from each other. The experimental results demonstrate that our method significantly outperforms the existing state-of-the-art methods across five benchmark datasets.

CCS CONCEPTS

• **Information systems → Information retrieval**; *Retrieval tasks and goals*; Information extraction.

KEYWORDS

Deep keyphrase generation; Neural hierarchical topic model; Variational neural generation model

ACM Reference Format:
Yuxiang Zhang, Tao Jiang, Tianyu Yang, Xiaoli Li, and Suge Wang. 2022. HTKG: Deep Keyphrase Generation with Neural Hierarchical Topic Guidance. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3531990

SIGIR '22, July 11–15, 2022, Madrid, Spain.

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531990

1 INTRODUCTION

Keyphrase prediction is to automatically produce a set of representative phrases that are related to the main topics discussed in a given document. Since keyphrases (also referred to as keywords) can provide a *high-level topic description* of a document, they are beneficial for a wide range of natural language processing (NLP) tasks, such as information extraction [67, 73], text summarization [69], opinion mining [7] and question answering [63]. However, the performance of existing approaches is still far from being satisfactory [31, 44]. The main reason is that it is very challenging to determine if a phrase or a set of phrases accurately capture the main topics (*i.e.*, salient information) that are presented in a document.

Automatic keyphrase prediction models can be broadly divided into traditional extraction and deep generation approaches. In particular, traditional extraction methods can only extract *present* keyphrases that appear in a given document, while deep generation methods can generate both present keyphrases as well as *absent* keyphrases that do not appear in the given document. In recent years, some topic-based methods for keyphrase prediction (including extraction and generation) have been proposed, mainly including topic-based extraction methods such as topic-based clustering methods [29, 45] and topical graph-based ranking methods [11, 12, 44, 64, 79, 83]. The work [70] is the only topic-based neural keyphrase generation method for short text on social media, which allows the joint learning of the latent flat topic representations. Although these topic-based methods have achieved promising results for the keyphrase prediction task, they all assume that topics are independent of one another and induce topics as *flat structures*, making generated keyphrases fall into a single topic (*i.e.*, generating duplicate/similar keyphrases).

In practice, a document usually covers different topics that are organized into a *hierarchical structure* rather than a *flat structure*. For example, we illustrate the corresponding hierarchical topic tree of this paper in Figure 1. Obviously, this topic tree has captured the underlying hierarchical document structure and associated semantics. Thus, it is extremely important to first learn the topic tree discussed in a given document, and subsequently leverage it to select most

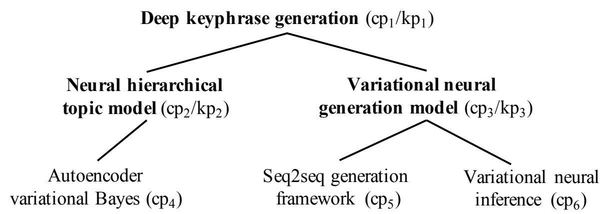

Figure 1: An example of hierarchical topic structure among topics of this paper, in which the topics are represented by corresponding candidate keyphrases (cp). The root conveys generic topic, while the leaves cover specific topics. Three candidate keyphrases with the high-level topic description are selected as keyphrases (kp) of this paper finally.

representative candidate keyphrases as the final keyphrases. This can ensure that the generated keyphrases cover all the major topics and provide high-level topic description. Unfortunately, to the best of our knowledge, none of the existing approaches leverage the hierarchical topic to guide the keyphrase generation.

This paper makes the first attempt to develop HTKG, a variational neural generation model guided by neural hierarchical topic to gain better performance in keyphrase generation. As illustrated in Figure 2, this method consists of two major modules: (1) a neural hierarchical topic model, and (2) a variational neural keyphrase generation model with hierarchical topic guidance. Specifically, the former aims to construct the latent topic tree across all the documents in a corpus (*i.e.*, compute a topic distribution over a tree for each word occurrence in a corpus) by adapting autoencoding variational Bayes (AEVB) framework [39]. The latter is designed to generate keyphrases with the designated hierarchical topic guidance by leveraging the sequence to sequence (seq2seq) generation model with the nonparametric variational neural inference. In the above two modules, we assume that the latent topic of an input document can be represented by a Gaussian mixture model (GMM), where each Gaussian component corresponds to a latent topic of the topic tree. Finally, these two modules are jointly trained to help them learn complementary information from each other.

Different from existing deep keyphrase generation approaches which directly encode from a source document and decode to its keyphrases, our proposed method introduces the latent variables to explicitly model underlying hierarchical topics of a source document and to guide the keyphrase generation via collaborative joint training of both the variational neural generation model and the neural hierarchical topic model. This makes our method more effective to capture the semantic hierarchical relations discussed in a document and thus generate keyphrases based on its semantic understanding with good topic coverage and accuracy. To summarize, our main contributions are as follows:

(1) To the best of our knowledge, this is the first attempt to leverage the neural hierarchical topics to guide deep keyphrase generation.
(2) We propose a novel hierarchical topic-guided variational neural keyphrase generation model, that not only effectively captures the long and strong dependencies between neighboring target words, but also utilizes the high-level topics discovered for keyphrase generation.
(3) We compare our HTKG method with three different types of existing methods, including seq2seq neural keyphrase generation methods, traditional keyphrase extraction methods and one graph neural keyphrase extraction method. Comprehensive experimental results demonstrate that our proposed method outperforms state-of-the-art baseline methods across five publicly-available datasets consistently.

The remaining part of this paper is organized as follows. We first summarize state-of-the-art approaches of keyphrase prediction and text generation via variational auto-encoders (VAEs) in Section 2. The proposed HTKG model is presented in Section 3. Finally, we introduce our datasets and experimental results in Section 4, before concluding the paper in Section 5.

2 RELATED WORK

2.1 Topic-based Keyphrase Extraction

The traditional extraction methods can be further classified into supervised and unsupervised approaches. In particular, supervised approaches treat keyphrase extraction as a binary classification task, using some classifiers, such as Naïve Bayes classifier [26, 46], boosted decision trees [59] and conditional random fields [2, 28]. In contrast, unsupervised approaches directly treat keyphrase extraction as a ranking problem, scoring each candidate using different kinds of unsupervised learning techniques, such as clustering [29, 45] and graph-based ranking [11, 12, 44, 50, 64, 67, 79, 83].

Topic information is used mainly in graph-based methods and most attempts involve biasing the ranking function towards topic distribution. Existing graph-based methods incorporating topic information induced by latent Dirichlet allocation (LDA) [10] include TopicalPageRank [44], cTPR [83], TPR [58], MIKE [79] and SalienceRank [64]. The other two works [11, 12] represent a given document as a multipartite graph of both topics and keyphrase candidates, and then select keyphrases from the top-ranked candidates, in which topics are defined as clusters of similar candidates. Nevertheless, in all these topic-based extraction methods, topics are *independent of one another* and organized as *flat structures.* In addition, compared with the newly developed generation methods, the traditional approaches suffer from poor performance [47].

2.2 Deep Keyphrase Prediction

CopyRNN [47] is the first to employ the attentional sequence to sequence (seq2seq) framework [62] with the copying mechanism [30] to generate both present and absent keyphrases for a document. Following this work, numerous extensions have been proposed to boost its generation ability. For instance, some studies incorporate different types of side information into seq2seq neural networks to improve keyphrase generation, such as correlation among keyphrases [17], title of source document [20], syntactic constraints [81] and topic information [70]. In addition, Ye et al., [74] propose a semi-supervised keyphrase generation model that utilizes both abundant unlabeled data and limited labeled data.

The above-mentioned early methods which use the standard seq2seq network can not generate multiple keyphrases and determine the appropriate number of keyphrases at a time for a target document. To overcome this drawback, Yuan et al., [78] introduce a new One2Seq training paradigm in the seq2seq network to generate

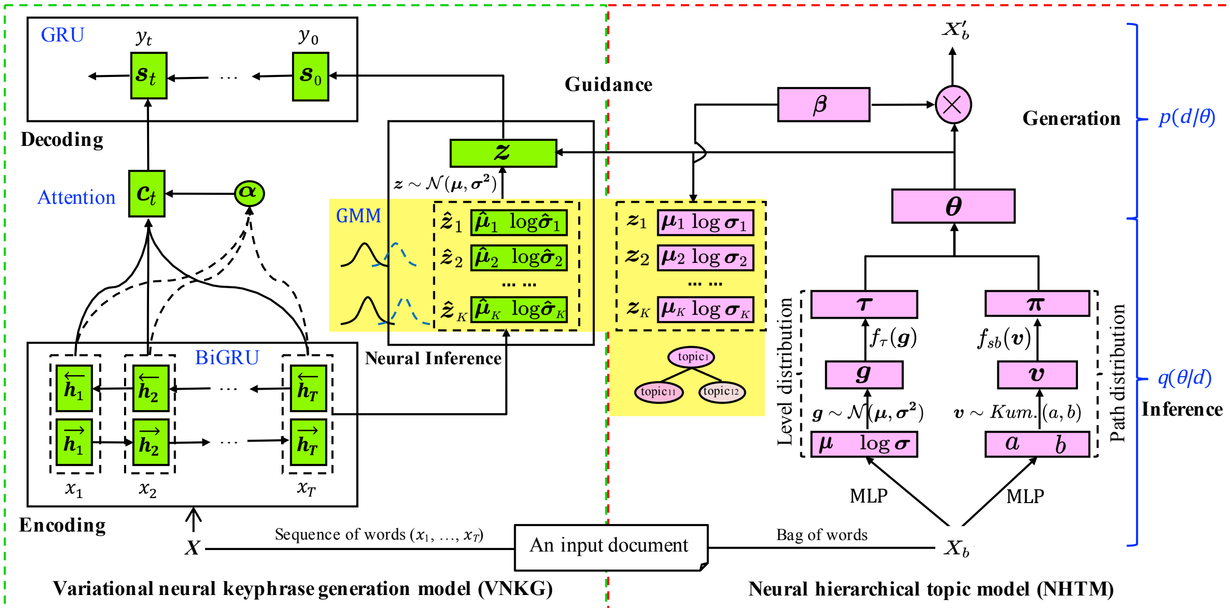

Figure 2: The overall architecture of the proposed HTKG model. This method consists of a *variational neural keyphrase generation model* with hierarchical topic guidance (left) to generate keyphrases, and a *neural hierarchical topic model* (right) to construct the topic tree as guidance signals. The two modules are jointly trained with an *inconsistency loss* (middle) to penalize the disagreement between the two hierarchical topic-guided Gaussian mixture distributions in VNKG and NHTM modules.

multiple keyphrases and decide the suitable number of keyphrases for a target document. Ye et al., [76] propose a One2Set paradigm to predict the keyphrases as a set, which eliminates the bias caused by the predefined order in One2Seq paradigm [78]. In addition, some works focus on improving the decoding process of seq2seq networks. Chen et al., [19] propose an exclusive hierarchical decoding framework and use either a soft or a hard exclusion mechanism to reduce duplication. Ahmad et al., [1] introduce an extractor-generator in the decoding to jointly extract and generate keyphrases from a target document. Bahuleyan et al., [4] adopt neural an unlikelihood objective to avoid generating duplicate keyphrases.

Besides the seq2seq networks (which can be implemented by the long short-term memory (LSTM) [32] or gated recurrent units (GRU) [22]), the neural graph-based networks, that extend traditional graph-based keyphrase ranking, have been used in keyphrase generation. Prasad et al., [55] firstly combine the advantages of traditional graph-based ranking methods and recent neural network-based approaches. Specifically, this method incorporates the global information (*i.e.*, TextRank ranking scores) into a graph attention network (GAT) [65] to extract keyphrases. Sun et al., [61] employ a graph convolutional neural network (GCN) [40] to encode the word graph into the corresponding representations and then adopt a pointer network [66] with diversity enabled attentions to generate keyphrases. Subsequently, Kim et al., [35] extend the word graph with structure information, and use GCN to extract the keyphrases for Web documents. Ye et al., [75] also enrich the word graph with related references and employ a GAT to generate the keyphrases.

We observe that almost all the existing deep keyphrase prediction approaches do not consider integrating the latent hierarchical topic information into the seq2seq framework to improve keyphrase prediction. In this paper, we first incorporate the hierarchical topical information into the variational neural sequence generation model, which can ensure that the generated keyphrases cover comprehensive topics and thus provide high-level topic description.

2.3 Text Generation via VAEs

Variational auto-encoders (VAEs) [39] are a type of deep generative models, which attempt to learn a compressed latent representation of the input by reconstructing the input data. VAEs have been extended by many following works in various specific language generation tasks, such as dialog generation [56, 82], text summarization [43, 72] and other natural language generation tasks [5, 49, 60, 77].

Additionally, several studies [23, 48, 57] directly use the VAE framework [39] to infer the topic distribution for words in a corpus, which improve data scalability comparing with probabilistic topic models such as LDA [10]. Besides these flat neural topic models, a few recent researches [21, 34, 54] reproduce the probabilistic hierarchical topic model nCRP [68], also using the VAE framework for improving data scalability. Different from these works, this paper attempts to integrate the hierarchical topic information into the neural sequence generation model for keyphrase generation.

3 METHODOLOGY: HTKG

3.1 Problem Definition and Framework

Given a corpus of documents $D=\{d_i\}_{i=1}^{|D|}$, where each document d ($d \in D$) is treated as a sequence of words $\boldsymbol{X}=(x_1,\ldots,x_{T_d})$ with length T_d, the goal of a keyphrase generation method is to find a model to generate a set of keyphrases $K = \{p_j\}_{j=1}^{|K|}$ for document d, where each keyphrase p can be treated as a sequence of words $\boldsymbol{Y} = (y_1, ..., y_{|p|})$. Note that as in existing deep text generation models, we use $\boldsymbol{X}$ and $\boldsymbol{Y}$ to denote the word sequence of an input document and the word sequence of its keyphrase, respectively.

The overall architecture of our proposed method is shown in Figure 2. It consists of two main modules: (1) *a neural hierarchical topic model* that computes a topic distribution over a tree for each word occurrence in a corpus, and (2) *a variational neural keyphrase generation model* to generate keyphrases with designated topic guidance. We jointly train them with an inconsistency loss so that they can learn complementary information from each other accurately. Below we first introduce the two main modules and then describe how they are jointly trained in detail.

3.2 Neural Hierarchical Topic Model (NHTM)

One innovation of this study is that it incorporates hierarchical topical information into keyphrase generation explicitly. Based on the current development of topic modeling, we follow the spirit of the neural hierarchical topic models [21, 34, 54] and adapt it to discover latent hierarchical topics. Figure 3 shows a topic tree in which each node is a topic. The topic at the root is the most general while topics at the leaf nodes are more specific. In this subsection, we first introduce the technical background and preliminaries and then describe the details of this model.

3.2.1 Technical Background and Preliminaries. Constructing a topic tree involves mainly two aspects: how to infer the latent topics in the text corpus, and how to organize these topics into a hierarchy. Traditional hierarchical topic models such as HLDA[9], nHDP [52] and rCRP [36], use conventional inference algorithms such as collapsed Gibbs sampling [8] and mean-field approximation [68], to infer the latent hierarchical topics. Current neural hierarchical topic models TSNTM [34], HTV [54] and nTSNTM [21], leverage the autoencoder variational Bayes framework, which can be trained together with neural networks and therefore has better adaptability and scales to large datasets.

To construct a topic tree with an infinite number of branches and levels, the existing methods follow the classical hierarchical LDA model nCRP [9, 68], which draws the *path* distribution from a nested stick-breaking construction as followings

$$v_k \sim \text{Beta}(1, \gamma),\ \pi_k = \pi_{par(k)} v_k \prod_{j=1}^{k-1} (1 - v_j), \tag{1}$$

and draws the *level* distribution from a stick-breaking construction as followings

$$\eta_l \sim \text{Beta}(1, \alpha),\ \theta_l = \eta_l \prod_{j=1}^{l-1} (1 - \eta_j), \tag{2}$$

where $k \in \{1, ..., K\}$ and $par(k)$ denote respectively the k-th topic and its parent. $l \in \{1, ..., L\}$ denotes the l-th level. v_k and η_l are stick proportions of topic k and level l, respectively.

3.2.2 Generative Process. Given a document d, we process it into a bag-of-words vector $\boldsymbol{X}_b \in \mathbb{Z}_+^{|V|}$, with $\mathbb{Z}_+$ denoting non-negative integers and V representing the vocabulary, in which each element reflects the number of times the corresponding word occurs in the document. To sample a topic for a word x_n in document d, a path c_n from the root to a leaf node and a level l_n are drawn. Let $\boldsymbol{\beta}_{c_n,l_n}$ be the topic distribution of the topic in path c_n and at level l_n. The full generative process of each word is given as follows

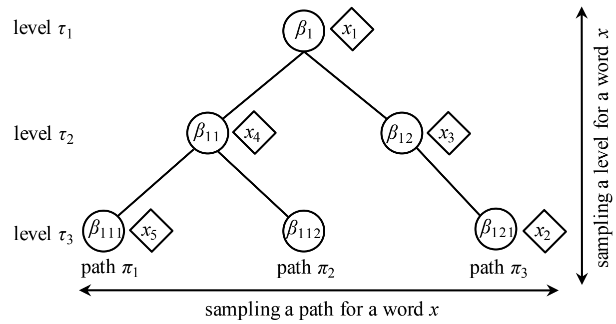

Figure 3: Sampling process of a topic for each word in a given document. For example, for word x_5, path π_1 and level τ_3 are sampled, and its assigned topic is β_{111}.

1. For a document d,

   Draw a breaking proportions: $\boldsymbol{v}_d \sim \text{Beta}(\alpha_0, \beta_0)$
   Obtain a path distribution: $\boldsymbol{\pi}_d = f_{sb}(\boldsymbol{v}_d)$
   Draw a Gaussian vector: $\boldsymbol{g}_d \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}^2)$
   Obtain a level distribution: $\boldsymbol{\tau}_d = f_\tau(\boldsymbol{g}_d)$

2. For a word x_n in document d,

   Draw a path: $c_n \sim \text{Mult}(\boldsymbol{\pi}_d),\ \text{for } n \in [1, N_d]$
   Draw a level: $l_n \sim \text{Mult}(\boldsymbol{\tau}_d),\ \text{for } n \in [1, N_d]$
   Draw a word: $x_n \sim \text{Mult}(\boldsymbol{\beta}_{c_n,l_n}),\ \text{for } n \in [1, N_d]$

where $f_{sb}(\cdot)$ is a stick-breaking construction function, and $f_\tau(\cdot)$ is a neural perceptron with softmax activation to transform a Gaussian sample to a level distribution.

3.2.3 Parameterizing Path Distribution and Level Distribution. Here we first parameterize the *path* distribution of document d. To bypass the obstacle that the Beta distribution does not have a differentiable non-centered parametrization that gradient-based inference requires [38], we approximate the Beta distribution by the Kumaraswamy distribution [42], which is a Beta-like distribution with a closed-form inverse cumulative distribution function and defined as Kumaraswamy$(x; a, b) = abx^{a-1}(1 - x^a)^{b-1}$ for $x \in (0, 1)$ and $a, b > 0$. Samples can be drawn via the inverse transform $x \sim (1 - u^{\frac{1}{b}})^{\frac{1}{a}}$, where $u \sim$ Uniform(0,1).

Given $\boldsymbol{\pi}_{l+1}$ which represents the path distribution of document d at level $l + 1$ ($\boldsymbol{\pi}_L = \boldsymbol{\pi}_d$), the path distribution at the upper level l can be inferred by

$$\boldsymbol{\pi}_l = \boldsymbol{\pi}_{l+1} \mathbf{M}_l \quad l = 1, ..., L - 1 \tag{3}$$

where $\mathbf{M}_l$ is a $K_{l+1} \times K_l$ matrix representing adjacent parent–child relationships between topics at level l and level $l+1$. Here, K_l is the number of topics at level l, and the item $\mathbf{M}_{l,i,j}$ denotes the degree of correlation between the i-th topic at level l and its j-th parent topic at level $l-1$ such that $\sum_j \mathbf{M}_{l,i,j} = 1$.

The exclusive posteriors a and b are estimated by the corresponding neural architectures $a = f_a(\boldsymbol{X}_b)$ and $b = f_b(\boldsymbol{X}_b)$, which are neural perceptrons with softplus activation, respectively.

Next, we parameterize the *level* distribution of document d. In particular, the exclusive posteriors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are estimated by the corresponding neural architectures $\boldsymbol{\mu} = f_\mu(\boldsymbol{X}_b)$ and $\boldsymbol{\sigma} = f_\sigma(\boldsymbol{X}_b)$, which are linear transformations respectively. In practice, we sample a $\tilde{\boldsymbol{g}}_d$ by employing the reparameterization trick [39], *i.e.*, $\tilde{\boldsymbol{g}}_d = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \tilde{\boldsymbol{\epsilon}}$ with the sample $\tilde{\boldsymbol{\epsilon}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}^2)$.

As in existing probabilistic topic models, we also use the latent variables $\boldsymbol{\theta}$ and $\boldsymbol{z}$ to denote respectively the topic proportion for a document and the topic assignment for the observed word x in the document, in which x is an element from the fixed vocabulary V. For a given document, its topic distribution $\boldsymbol{\theta}$ is associated with a *path* distribution $\boldsymbol{\pi}$ over all the paths from the root to the leaf nodes, and a *level* distribution $\boldsymbol{\tau}$ over all tree levels. More formally, the topic distribution $\boldsymbol{\theta}$ of document d can be defined as $\boldsymbol{\theta} = \{\boldsymbol{\theta}_l\}_{l=1}^{L}$ (where L is the depth of the topic tree), which is the topic distribution of document d, derived as

$$\boldsymbol{\theta}_l = \tau_l \boldsymbol{\pi}_l, \quad l = 1, ..., L \tag{4}$$

3.2.4 Parameterizing Topic-word Distribution. Here we follow the work [48] to explicitly compute topic-word distribution (*i.e.*, word distribution assigned to topic k) by

$$\boldsymbol{\beta}_k = \mathrm{softmax}(\mathbf{E}_w \cdot \boldsymbol{e}_k^\top) \tag{5}$$

where $\boldsymbol{e}_k$ is the embedding of topic k, and $\mathbf{E}_w$ is the embeddings of all words.

3.2.5 Hierarchical Topic-guided Gaussian Mixture Prior. Given the topic-word distribution $\boldsymbol{\beta}$ and the topic distribution $\boldsymbol{\theta}$ of a document, the hierarchical topic-guided Gaussian mixture is

$$p(z|\boldsymbol{X}_b) = \sum\nolimits_{k=1}^{K} \theta_k(\boldsymbol{X}_b) \mathcal{N}\left(z_k; \boldsymbol{\mu}_k(\boldsymbol{X}_b), \boldsymbol{\sigma}_k^2(\boldsymbol{X}_b)\right) \tag{6}$$

where the mean $\boldsymbol{\mu}_k$ and standard derivation $\boldsymbol{\sigma}_k$ are obtained by the fully connected layer as

$$\begin{aligned} \boldsymbol{\mu}_k &= \mathbf{W}_\mu \boldsymbol{\beta}_k + \boldsymbol{b}_\mu \\ \log \boldsymbol{\sigma}_k &= \mathbf{W}_\sigma \boldsymbol{\beta}_k + \boldsymbol{b}_\sigma \end{aligned} \tag{7}$$

where $\boldsymbol{\beta}_k$ is the word distribution assigned to topic k. Unlike a normal GMM prior that sets each mixture component to be $\mathcal{N}(0, \boldsymbol{I}^2)$, this topic-guided GMM provides the topic information for each mixture component, thus making the model more interpretable for keyphrase generation.

3.2.6 Variational Inference. Given topic-word distribution $\boldsymbol{\beta}$ and topic distribution $\boldsymbol{\theta}$ of a given document d, this document is reconstructed and its marginal likelihood is

$$\begin{aligned} p(\boldsymbol{X}_b|\boldsymbol{\beta}) &= \int_{\boldsymbol{\pi},\boldsymbol{\tau}} \Big\{ \prod_n \sum_{c_n, l_n} p(x_n|\boldsymbol{\beta}_{c_n,l_n}) p(c_n|\boldsymbol{\pi}) p(l_n|\boldsymbol{\tau}) \Big\} \\ &\qquad p(\boldsymbol{\pi}) p(\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\pi} \mathrm{d}\boldsymbol{\tau} \\ &= \int_{\boldsymbol{\theta}} \Big\{ \prod_n \sum_{z_n} p(x_n|\boldsymbol{\beta}_{z_n}) p(z_n|\boldsymbol{\theta}) \Big\} p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \\ &= \int_{\boldsymbol{\theta}} \Big\{ \prod_n (\boldsymbol{\theta} \cdot \boldsymbol{\beta})_{x_n} \Big\} p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \end{aligned} \tag{8}$$

Based on the AEVB framework, the evidence lower bound for the document log-likelihood is derived as

$$\begin{aligned} \mathcal{L}_{ht} =& \mathbb{E}_{q(\boldsymbol{\pi},\boldsymbol{\tau}|\boldsymbol{X}_b)} \Big[\sum_n \log(\boldsymbol{\theta} \cdot \boldsymbol{\beta})_{x_n} \Big] \\ & - \mathrm{KL}\big[q(\boldsymbol{\pi}|\boldsymbol{X}_b) || p(\boldsymbol{\pi})\big] - \mathrm{KL}\big[q(\boldsymbol{\tau}|\boldsymbol{X}_b) || p(\boldsymbol{\tau})\big] \end{aligned} \tag{9}$$

where $q(\boldsymbol{\pi}|\boldsymbol{X}_b)$ and $q(\boldsymbol{\tau}|\boldsymbol{X}_b)$ are posteriors modeled by the inference network, and the priors $p(\boldsymbol{\pi})$ and $p(\boldsymbol{\tau})$ are given in the previous *Generative Process* subsection.

3.3 Variational Neural Keyphrase Generation (VNKG) Guided by Hierarchical Topic

Different from traditional seq2seq keyphrase generation methods such as CopyRNN [47] and SEG-Net [1], our keyphrase generation model is a variational sequence generation model, based on the seq2seq framework model and the variational neural inference (VNI) [13, 39, 60]. Specifically, we introduce a latent variable, which is guided by the hierarchical topic model described in the previous section, to model the underlying topic space as a global signal for keyphrase generation. Thus, it should be able to capture the high-level topic in a given document.

3.3.1 Variational Neural Encoder. This module aims at encoding an input document into continuous vectors. Let $\boldsymbol{X} = (x_1, ..., x_T)$ be a sequence of words within an input document, and $\boldsymbol{x} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_T]$ be its corresponding sequence of word embeddings. We adopt a bidirectional gated recurrent unit (BiGRU) [3] as the encoder, which maps the input word sequence $\boldsymbol{X}$ into a set of contextualized hidden states $\boldsymbol{h} = [\boldsymbol{h}_1, ..., \boldsymbol{h}_T]$ as

$$\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T = \mathrm{BiGRU}(\boldsymbol{x}_1, \boldsymbol{x}_1, ..., \boldsymbol{x}_T). \tag{10}$$

In this way, each contextualized vector $\boldsymbol{h}_i$ encodes information about the i-th word with respect to all the other surrounding words in the sequence. The last hidden state of the encoder $\boldsymbol{h}_T$ is used to calculate the latent topic variable $\boldsymbol{z}$.

3.3.2 Hierarchical Topic-guided Gaussian Mixture Posterior. In this model, we consider incorporating the topic information into latent variables. Each topic is drawn from a topic-dependent multivariate Gaussian distribution, computed as

$$p(z|\boldsymbol{X}) = \sum\nolimits_{k=1}^{K} \theta_k(\boldsymbol{X}_b) \mathcal{N}\left(\hat{z}_k; \hat{\boldsymbol{\mu}}_k(\boldsymbol{X}), \hat{\boldsymbol{\sigma}}_k^2(\boldsymbol{X})\right) \tag{11}$$

where θ_k is the usage of topic k in a document, computed by our NHTM model. To estimate $\hat{z}_k$, we introduce the fully connected layer to obtain vectors $\hat{\boldsymbol{\mu}}_k$ and $\log\hat{\boldsymbol{\sigma}}_\mathrm{k}$ as follows

$$\begin{aligned} \hat{\boldsymbol{\mu}}_k &= \mathbf{W}_{\mu_k} \boldsymbol{h}_T + \boldsymbol{b}_{\mu_k} \\ \log \hat{\boldsymbol{\sigma}}_k &= \mathbf{W}_{\sigma_k} \boldsymbol{h}_T + \boldsymbol{b}_{\sigma_k} \end{aligned} \tag{12}$$

Finally, to obtain a representation for the latent topic variable $\boldsymbol{z}$, we follow the reparameterization trick of VAE to implement it.

3.3.3 Variational Neural Decoder. Given a source document $\boldsymbol{X}$ and a continuous latent topic variable $\boldsymbol{z}$, the process to generate its keyphrase Y is defined as following conditional probability

$$p(Y|\boldsymbol{X}) = \prod\nolimits_{t=1}^{|Y|} p(y_t|Y_{<t}, \boldsymbol{z}, \boldsymbol{X}) p(\boldsymbol{z}|\boldsymbol{X}) \tag{13}$$

where $Y_{<t} = (y_1, ..., y_{t-1})$ is a previously generated word sequence.

Table 1: Summary of the training, validation and testing datasets.

Dataset		#Abs	#PKps	%PKps	#AKps	%AKps	#Avg.Kps	#Avg.PKps	#Avg.AKps
Train.	KP20k	1,709,490	994,880	63.2	513,918	36.8	5.26	3.32	1.94
Valid.	KP20k	19,992	66,355	63.1	38,772	36.9	5.25	3.31	1.94
Test.	Inspec	500	3,602	73.6	1,293	26.4	9.79	7.20	2.59
	Krapivin	400	1,297	55.6	1,037	44.4	5.84	3.24	2.59
	NUS	211	1,191	52.2	1,088	47.8	10.8	5.65	5.15
	SemEval	100	612	42.4	831	57.6	14.43	6.12	8.31
	KP20k	20000	66,267	62.9	39,076	37.1	5.26	3.31	1.95

The decoder is another forward GRU, which is used to generate the sequence of keyphrases by predicting the next word y_t based on the hidden state $\boldsymbol{s}_t$ of the decoder at timestep t. Both y_t and $\boldsymbol{s}_t$ are conditioned on y_{t-1} and $\boldsymbol{c}_t$ of the input sequence. Formally, the hidden state $\boldsymbol{s}_t$ and decoding function can be written as

$$\boldsymbol{s}_t = \mathrm{GRU}_f(y_{t-1}, \boldsymbol{s}_{t-1}, \boldsymbol{c}_t) \tag{14}$$

and

$$p(y_t|Y_{<t}, \boldsymbol{z}, \boldsymbol{X}) = g(y_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}_t) \tag{15}$$

where $\boldsymbol{c}_t = \sum_i \alpha_{ti}\boldsymbol{h}_i$ is a source context vector computed as the weighted sum of the source hidden states $\{\boldsymbol{h}_i\}$ using the attention mechanism [3], and $g(\cdot)$ is a nonlinear multi-layered function that outputs the probability of y_t. The latent topic variable $\boldsymbol{z}$ is used to initialize the hidden state $\boldsymbol{s}_0$ in the decoder.

Finally, we minimize the cross entropy loss function to train this generation model

$$\mathcal{L}_{kg} = -\sum_{i=1}^{N} \log\left(p(y_i|\boldsymbol{X}, Y_{<i}, \boldsymbol{z})\right) \tag{16}$$

where N denotes the length of target keyphrases, and $\boldsymbol{z}$ is the latent topic of the given document.

3.4 Joint Learning

Since keyphrase generation and topic modeling both aim to distill salient information from input documents, we jointly train the two modules to help them learn complementary information from each other. The loss function of our model consists of three parts. Two of them, namely, hierarchical topic loss $\mathcal{L}_{ht}$ and keyphrase generation loss $\mathcal{L}_{kg}$, have been given in the previous subsections.

To push the hierarchical topic-guided Gaussian mixture computed in VNKG towards the corresponding distribution computed in NHTM, we devise the third loss — an inconsistency loss $\mathcal{L}_{ic}$. For these two mixture Gaussian distributions $p(\boldsymbol{z}|\boldsymbol{X}_b) = \sum_{i=1}^{K} \theta_i \mathcal{N}(\mu_i, \sigma_i^2)$ and $p(\boldsymbol{z}|\boldsymbol{X}) = \sum_{i=1}^{K} \theta_i \mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i^2)$, their Kullback–Leibler (KL) divergence is upper-bounded by

$$\begin{aligned} \mathcal{L}_{ic} &= \mathrm{KL}\left(p(\boldsymbol{z}|\boldsymbol{X}_b)||p(\boldsymbol{z}|\boldsymbol{X})\right) \\ &\leq \mathrm{KL}(\boldsymbol{\theta}||\boldsymbol{\theta}) + \sum_{i=1}^{K} \theta_i \mathrm{KL}\left(\mathcal{N}(\mu_i, \sigma_i^2)||\mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i^2)\right) \end{aligned} \tag{17}$$

where $\mathrm{KL}(\boldsymbol{\theta}||\boldsymbol{\theta})$ is equal to 0. The general form of this formula has been proven to be correct in the study [24].

The final overall loss of the entire framework's training objective is the linear combination of the three parts, defined as

$$\mathcal{L} = \mathcal{L}_{ht} + \mathcal{L}_{kg} + \mathcal{L}_{ic}. \tag{18}$$

4 EXPERIMENTS

4.1 Datasets

We employ the dataset KP20k collected by Meng et al., [47], which contains a large amount of high-quality scientific metadata in the computer science domain from various online digital libraries. In this dataset, each example contains a title and an abstract of a scientific publication as source text, and multiple author-assigned keywords as target keyphrases. Following previous works [47, 78], we split this dataset into training, validation and test sets, and use the training set to train all the deep seq2seq models. We use the validation set to find the optimal hyperparameters during the training process. Finally, we apply our models in the test set and report their performance.

In order to evaluate the proposed model comprehensively, we also test the model trained with KP20k on other four widely-adopted public datasets from the scientific domain, namely, Inspec [33], Krapivin [41], SemEval-2010 [37] and NUS [51]. The detailed statistic information of these five benchmarks are summarized in Table 1, along with the number of abstracts (#Abs), the number of and the percentage of present keyphrases (#PKps and %PKps), the number of and the percentage of absent keyphrases (#AKPs and %AKps), and the average number of keyphrases, present and absent keyphrases per document (#Avg.Kps, #Avg.PKps and #Avg.AKps).

4.2 Comparative Methods

To comprehensively evaluate the performance of our HTKG[1], we compare our method with two types of methods, including nine conventional *keyphrase extraction* methods and seven *deep keyphrase generation* methods. Conventional extraction methods consist of three different types:

- *Statistic-based* unsupervised methods include (1) **TF-IDF** which directly ranks each candidate word according to its TF-IDF score (*i.e.*, the term frequency-inverse document frequency), and (2) **YAKE!** [15] which is a recently proposed method and computes a combined score based on five word features, such as casing aspect, frequency and position.
- *Graph-based* unsupervised methods include the following four models. (3) **TextRank** [50] is the first to use the PageRank algorithm on the word graph to rank candidate words. (4) **SingleRank** [67] is a TextRank extension by adding a few neighbor documents close to the target document. (5) **PositionRank** [25] is also a TextRank extension by incorporating

[1]The code of our model is available in public at https://github.com/HDKG/HTKG.

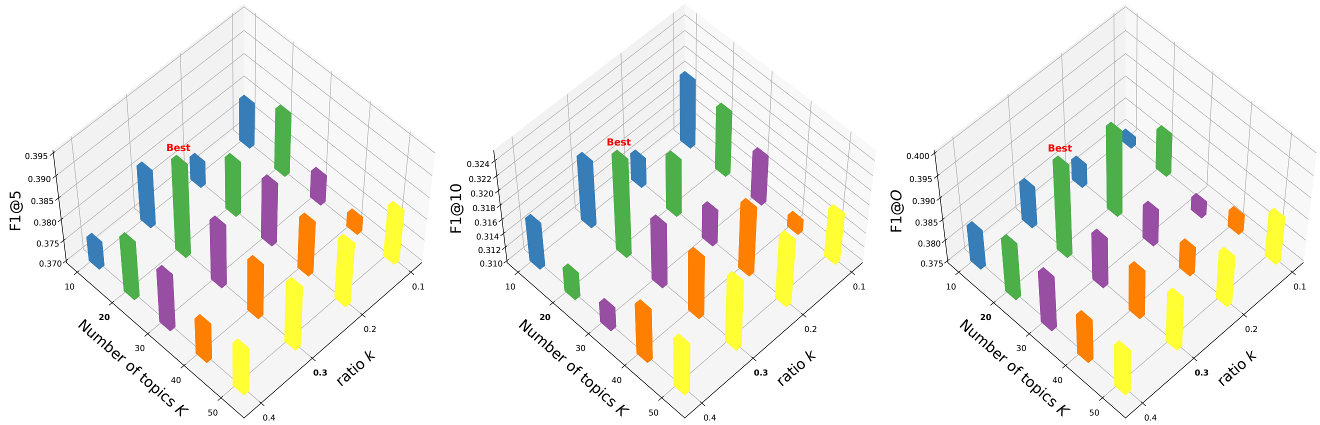

Figure 4: The influence of the structure of the topic tree on KP20k dataset.

the positional information of a word's occurrences into a biased PageRank. (6) **KPRank** [53] also extends TextRank by exploiting both positional information and contextual word embeddings into a biased PageRank.

- Traditional supervised methods include (7) **KEA** [71] which employs only two features (TF-IDF and relative position) and applies Naïve Bayes classifier, and (8) **Maui** [46] which chooses nine features and applies bagged decision trees [14] to determine whether a word is a keyword or not.

Due to the limited space, we select the *best-performing baseline* (BL*) from each of the three types of baselines with the best-performing metrics to compare with our method for each dataset. A current graph neural extraction method is given as

- (9) **DivGraphPointer** [61] employs a GCN encoder on the traditional word graph for keyphrase extraction.

The seven current deep seq2seq generation baselines include:

(1) **CopyRNN** [47] is the first to use seq2seq network to generate keyphrases. Here, we replace it with CopyRNN$^+$ which is re-implemented CopyRNN with best results [78].
(2) **CopyCNN** [80] applies a convolutional neural network based encoder-decoder framework to generate keyphrases.
(3) **KG-KE-KR-M** [18] is a multi-task learning method using extractive and generative models to generate keyphrases.
(4) **CatSeq** [78] has the same framework as CopyRNN, with the key difference in training paradigm.
(5) **CatSeqTG-2RFl** [16] is a simple extension of CatSeq using reinforcement learning to generate both sufficient and accurate keyphrases.
(6) **ExHiRD-h** [19] uses an exclusive hierarchical decoder to avoid generating duplicated keyphrases.
(7) **One2Set** [76] is a new training paradigm without predefining an order to concatenate the keyphrases.

4.3 Evaluation Metrics

For fairly comparing different approaches, we follow the literature and adopt top-N macro-averaged *precision*, *recall* and F_1-*measure* as the evaluation metrics. In particular, precision is defined as the number of correctly predicted keyphrases over the number of all predicted keyphrases, recall is defined as the number of correctly predicted keyphrases over the total number of data records, and F_1 is the harmonic mean of precision and recall.

Note $F_1@k$ is used in almost all existing works on the keyphrase extraction and generation, in which k (usually 5 or 10) is a fixed number of top-k predictions. $F_1@O$ is recently proposed in the work [78] as one of our evaluation metrics, in which O is the number of author-assigned keyphrases. This means that the number of predicted phrases taken for evaluation is the same as the number of ground truth keyphrases for each document. The recall of the top 10 predictions ($R@10$) is used to evaluate the performance of methods for predicting absent keyphrases.

4.4 Experimental Setup

We follow the previous works [47, 78] to pre-process the experimental data, including lowercasing, tokenizing, etc. Particularly, the top 50,000 and 10,000 most frequently-occurred words in the training data are selected as the vocabulary shared in the sequence encoder and decoder, and as the bag-of-words vocabulary in the neural hierarchical topic model, respectively.

For the neural hierarchical topic model, we set the size of hidden layers to 256 and use one sample for neural variational inference by following the work [48]. The parameters α_0 and β_0 for the Beta distribution are empirically set to 1 and 10, respectively.

For the neural keyphrase generation model, the word embeddings are initialized first using normal distribution by the method [27], and the size of word embedding is set as 150. The size of hidden state of Bi-GRU encoder is set as 150, and the size of hidden state of forward GRU decoder is set as 300.

In the training process, we adopt One2One training paradigm [47] and use Adam [6] as optimizer to optimize all the parameters. The initial learning rate is set as 0.001 and the gradient clipping is set as 1. The batch sizes of the topic model and the keyphrase generation model are set to 1024 and 128, respectively. We halve the learning rate when the validation performance drops, and stop training if it does not improve for three successive iterations. In addition, we pre-train the hierarchical topic model for 100 epochs before the

Table 2: Results of predicting *present* keyphrases of different methods on five datasets. Best/second-best performing score in each column is highlighted with bold/underline. Gain reports the relative improvements between our HTKG and the best/second-best performance results in the same deep generation methods.

Model	**KP20k**			Inspec			Krapivin			NUS			SemEval		
	F_1@5	F_1@10	F_1@O	F_1@5	F_1@10	F_1@O	F_1@5	F_1@10	F_1@O	F_1@5	F_1@10	F_1@O	F_1@5	F_1@10	F_1@O
Seq2seq neural *generation* methods															
CopyRNN+ [78]	31.7	27.3	33.5	24.4	28.9	29.0	30.5	26.6	32.5	37.6	35.2	40.6	31.8	**31.8**	31.7
CopyCNN [80]	35.1	28.8	-	28.5	34.6	-	31.4	27.2	-	34.2	33.0	-	29.5	30.8	-
KG-KE-KR-M [18]	31.7	28.2	38.8	25.7	28.4	31.4	27.2	25.0	31.7	28.9	28.6	38.4	20.2	22.3	30.3
CatSeq [78]	31.4	27.3	31.9	29.0	30.0	30.7	30.7	**27.4**	32.4	35.9	34.9	38.3	30.2	30.6	**31.0**
CatSeqTG-2RF1 [16]	32.6	20.2	35.7	26.6	18.1	22.4	31.2	19.3	**34.7**	36.5	24.4	39.6	27.7	19.0	25.5
ExHiRD-h [19]	31.1	19.6	37.4	25.3	18.3	28.9	28.4	18.0	30.6	-	-	-	29.2	20.3	26.6
One2Set [76]	35.5	23.7	36.9	28.2	20.8	25.4	31.5	20.8	34.3	39.7	28.2	39.5	**34.0**	24.2	30.2
HTKG (This work)	**39.1**	**32.3**	**39.4**	**32.4**	**34.9**	**33.8**	**32.3**	26.3	31.8	**42.2**	**39.1**	**42.8**	32.5	31.4	30.8
Gain	***3.6***$\uparrow$	***3.5***$\uparrow$	*0.6*$\uparrow$	***3.4***$\uparrow$	*0.3*$\uparrow$	***2.4***$\uparrow$	*0.8*$\uparrow$	*-1.1*$\downarrow$	*-2.9*$\downarrow$	***2.5***$\uparrow$	***3.9***$\uparrow$	*2.2*$\uparrow$	*-1.5*$\downarrow$	*-0.4*$\downarrow$	*-0.2*$\downarrow$
Traditional *extraction* methods (unsup. denotes unsupervised.)															
Statistic-unsup. BL*	14.1	14.6	6.3	20.4	26.9	24.8	21.5	19.6	13.3	15.9	19.6	12.5	15.1	21.2	15.3
Graph-unsup. BL*	18.1	15.1	18.4	28.6	33.9	33.5	18.5	16.0	21.1	23.0	21.6	23.8	22.5	25.7	22.9
Supervised BL*	4.6	4.4	5.1	9.8	12.6	3.9	11.0	15.2	1.7	6.9	8.4	8.1	6.8	6.5	6.6
Graph neural *extraction* method (Note that extraction methods can not predict absent keyphrases)															
DivGraphPointer [61]	36.8	29.2	-	38.6	41.7	-	46.0	40.2	-	40.1	38.9	-	36.3	29.7	-

joint training as the convergence speed of our VNKG is much faster than our NHTM. To alleviate the problem that the posterior collapse issue in VAE framework, where the decoder tends to ignore the latent variables, we employ the simple KL cost annealing technique [13]. More specifically, we add a variable weight to the KL term in the loss function at training time. At the start of training, we set that weight to 0, and then we gradually increase this weight to 1 as the training progresses. In the testing process, our models use the beam search with a width of 200 and a max depth of 6.

4.5 Influence of Topic-Tree Structures

In our NHTM model, we construct a three-level topic tree, in which we fix one root topic at level 1 to capture the the highest-level topic (*i.e.*, generic topic) discussed in a document. To illustrate the influence of the structure of the topic tree (*i.e.*, width and depth), we vary the number of total topics K in the range of 10 to 50, and the ratio of the number of level-2 topics to total of topics k in the range of 0.1 to 0.4. The empirical upper bound of 0.4 for k is based on the assumption that the number of level-3 topics is more than that of level-2 topics. Here we adopt these two intuitive parameters (*i.e.*, K and k) to replace the width and depth of the topic tree, which can be computed from the former.

The results of F_1@5, F_1@10 and F_1@O on KP20k dataset are shown in Figure 4. From this figure, we observe that the performance of HTKG is obviously influenced by changes on both the number of topics K and the ratio k. In general, the F_1@5, F_1@10 and F_1@O *quickly* increase and then slowly decrease on KP20k dataset as K grows, respectively. The F_1@5, F_1@10 and F_1@O *slowly* increase and then quickly decrease on KP20k dataset as k grow, respectively. The best-performing setting is the number of total topics $K=20$ and the ratio $k=0.3$ on KP20k dataset, which is finally used in the comparison experiments. In practice, the best-performing structure of the topic tree is related to the corpus.

4.6 Performance Comparison

We compare HTKG with the baselines on five datasets, and the experimental results for present and absent keyphrase prediction are shown in Table 2 and 3, respectively. Note that Table 2 also includes a set of extraction methods, which can only extract present keyphrases. That is to say, they are not quite suitable for directly comparing different types of prediction methods. In addition, due to space limitations and metric specialization, we present only results obtained with the most suitable metrics for each type of methods. Specifically, we choose F_1@5, F_1@10 and F_1@O for the present methods and F_1@5, F_1@O and R@10 for the absent methods.

4.6.1 Present keyphrase prediction. From the results of predicting *present* keyphrases illustrated in Table 2, we can see that the neural prediction methods (including the seq2seq neural generation methods and the graph neural extraction method DivGraphPointer) substantially outperform the traditional extraction methods across all the datasets. This improvement benefits from the deep semantic understanding of a document.

The results also show that our HTKG outperforms all the seq2seq baseline methods by significant margins in three out of five datasets (including KP20k, Inspec and NUS) in terms of all the metrics. Specifically, HTKG achieves the improvement of 3.6 F_1@5 points and 3.5 F_1@10 points on KP20k over the best baselines, of 3.4 F_1@5 points and 2.4 F_1@O points on Inspec, and of 2.5 F_1@5 points and 3.9 F_1@10 points on NUS, respectively. These results illustrate HTKG can achieve the average increase of 3 points on these metrics, which is a significant improvement in the current keyphrase prediction task. On both Krapivin and SemEval datasets, HTKG performs

Table 3: Results of generating *absent* keyphrases of different methods on five datasets.

Model	KP20k			Inspec			Krapivin			NUS			SemEval		
	$F_1@5$	$F_1@O$	$R@10$	$F_1@5$	$F_1@O$	$R@10$	$F_1@5$	$F_1@O$	$R@10$	$F_1@5$	$F_1@O$	$R@10$	$F_1@5$	$F_1@O$	$R@10$
Deep generation methods															
CopyRNN+ [78]	3.23	3.97	3.30	1.44	1.23	4.00	5.14	5.54	4.00	3.35	3.25	2.40	2.05	2.20	0.90
CatSeq [78]	1.50	2.51	6.00	0.40	0.35	2.90	1.80	1.67	7.00	1.60	1.22	3.70	1.60	1.44	2.50
CatSeqTG-2RF1 [16]	2.78	2.16	4.70	1.12	0.82	1.69	2.97	1.88	4.38	2.46	2.10	2.57	2.04	1.78	1.77
ExHiRD-h [19]	1.57	3.22	2.61	1.03	1.52	1.84	2.06	2.49	3.09	–	–	–	1.51	1.14	1.30
One2Set [76]	3.70	5.02	6.08	2.10	1.63	3.63	5.53	3.43	7.93	4.02	3.61	4.96	2.42	2.02	2.02
HTKG (This work)	**5.85**	**6.04**	**13.2**	**2.20**	**1.83**	**5.38**	**5.64**	**5.68**	**12.2**	**5.06**	**5.03**	**10.2**	**3.25**	**3.60**	**4.02**
Gain	*2.2↑*	*1.0↑*	***7.1↑***	*0.1↑*	*0.2↑*	*1.4↑*	*0.1↑*	*0.1↑*	***4.3↑***	*1.0↑*	*1.4↑*	***5.2↑***	*0.8↑*	*1.4↑*	*1.5↑*

slightly worse than the best baselines. This slight performance drop may be caused by the various topics discussed in the given datasets. For example, the selected articles in SemEval dataset belong to both computer science and economics domains.

Even aside from generating absent keyphrases, HTKG outperforms DivGraphPointer (which is specially designed to extract present keyphrases and can not predict absent keyphrases) on two out of five datasets by a significant margin (2.3 $F_1@5$ points and 3.1 $F_1@10$ points on KP20k, and 2.1 $F_1@5$ points and 0.2 $F_1@10$ points on NUS). In short, unlike all previous methods, HTKG integrates the hierarchical topics into keyphrase generation explicitly, contributing most to the performance improvement on this task.

4.6.2 Absent keyphrase prediction. Unlike present keyphrases, absent keyphrases do not appear in the target document, and thus predicting them is very challenging and requires comprehensive understanding the latent document semantic. From the results of predicting *absent* keyphrases presented in Table 3, we can see that HTKG substantially outperforms the baselines according to all the metrics, and correctly generates more absent keyphrases than the baselines on the five datasets consistently in terms of $R@10$, especially on KP20k (7.1 $R@10$ points over the best existing methods), Krapivin (4.3 $R@10$ points) and NUS (5.2 $R@10$ points). Besides, HTKG doubles the average $R@10$ on these three datasets, compared to the best existing methods. Overall, the absent keyphrase prediction results indicate that HTKG is capable of understanding the underlying document semantic better than all the baselines, and thus generating much better results.

4.7 Ablation Study

To analyze the relative contributions of different components to the model performance in predicting present and absent keyphrases, we compare our full model HTKG with the following ablated variants: (1) *w/o hierarchical topic* where the hierarchical topic model is replaced by the flat topic model NTM [48], (2) *w/o VNI for topics*, where we directly concatenate the topic representations learned by NTM [48] and last hidden state of the sequential encoder, and feed into the decoder to generate keyphrases.

From the results shown in Table 4, we have the following observations: (1) Replacing the *hierarchical topics* with the *flat topics* leads to performance drops on all datasets, indicating that the hierarchical topic is effective information to improve keyphrase generation. (2) The simple concatenation results in significant performance drop on all datasets. However, compared to the earliest baseline CopyRNN, slight improvements of the performance are observed in conjunction with the results shown in Table 2 and Table 3. These results indicate that although the simple concatenation contributes to improving the performance, this option can not effectively leverage the topic information to guide the keyphrase generation.

Table 4: Ablation on HTKG without decoupling hierarchical topic (HTopic) and VNI for topics (VNI). We preclude one design choice at a time.

Dataset	Method	Present		Absent	
		$F_1@5$	$F_1@O$	$F_1@5$	$F_1@O$
KP20k	HTKG	39.1	39.4	5.85	6.04
	w/o HTopic	36.5	37.1	5.13	5.62
	w/o VNI	31.8	33.6	3.25	3.93
Inspec	HTKG	32.4	33.8	2.20	1.83
	w/o HTopic	30.1	31.9	1.87	1.44
	w/o VNI	24.5	29.1	1.46	1.25
Krapivin	HTKG	32.3	31.8	5.64	5.68
	w/o HTopic	31.7	30.9	5.47	5.52
	w/o VNI	30.6	32.5	5.22	5.53
NUS	HTKG	42.8	39.6	5.06	5.03
	w/o HTopic	40.2	37.7	4.31	4.14
	w/o VNI	37.3	34.6	3.36	3.15
SemEval	HTKG	32.5	30.8	3.25	3.60
	w/o HTopic	32.2	30.4	2.83	3.01
	w/o VNI	31.8	29.7	2.05	2.20

4.8 Case Study

A anecdotal example is shown in Table 5. The given paper focuses on developing a new approach to “algebraic attack”, which is an author-assigned *absent* keyphrase and appears in the first position of keyphrase sequence. Obviously, the keyphrase “algebraic attack” can be regarded as the root topic description, and the rest of keyphrases are derived from this root keyphrase. Under the hierarchical topic guidance, HTKG is capable of understanding the underlying document semantic, and thus can accurately generate this absent root keyphrase while all the baselines fail to generate it. Besides, we also empirically observe more keyphrases with less reputation are generated by HTKG.

Table 5: An example of generated keyphrases by different methods. Author-assigned (*i.e.*, Gold) keyphrases are shown in bold, and absent keyphrases are labeled by the underline.

"New constructions of even variable rotation symmetric boolean functions with maximum algebraic immunity" (#4434 in KP20k; $F_1@O$)

Gold: **algebraic attack**; **nonlinearity**; **rotation symmetry**; **boolean function**; **algebraic immunity**

HTKG: *cryptography*; ***algebraic attack***; ***boolean functions; nonlinearity***; ***algebraic immunity***

CatSeq: **algebraic immunity**; maximum algebraic immunity; rotation symmetric boolean functions

CatSeqTG-2RF1: **boolean functions**; **algebraic immunity**; rotation symmetric boolean functions; **rotation symmetry**

ExHiRD-h: even variable rotation symmetric boolean; **algebraic immunity**; **boolean function**; symmetric boolean function

One2Set: rotation symmetric boolean functions; cryptography; **boolean function**; maximum algebraic immunity; even variable

5 CONCLUSION

In this study, we propose a hierarchical topic-guided variational neural keyphrase generation method, which incorporates the hierarchical topic information into keyphrase generation explicitly. In particular, we jointly learn both latent hierarchical topics and keyphrases, allowing our model to better exploit the mutual reinforcement between them, and accurately capturing the topics and relations between them discussed in a given document. We conducted comprehensive experiments to demonstrate its advantages and effectiveness. In future, we plan to evaluate HTKG on a large corpus with comprehensive coverage of diverse topics.

ACKNOWLEDGMENTS

This work was partially supported by grants from the National Natural Science Foundation of China (Grant No. U1933114, 62076158), and Scientific Research Project of Tianjin Educational Committee (Grant No. 2021ZD002).

REFERENCES

[1] Wasi Uddin Ahmad, Xiao Bai, Soomin Lee, and Kai-Wei Chang. 2021. Select, Extract and Generate: Neural Keyphrase Generation with Layer-wise Coverage Attention. In *Proceedings of ACL*.

[2] Rabah Alzaidy, Cornelia Caragea, and C Lee Giles. 2019. Bi-LSTM-CRF sequence labeling for keyphrase extraction from scholarly documents. In *Proceedings of WWW*.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

[4] Hareesh Bahuleyan and Layla El Asri. 2020. Diverse keyphrase generation with neural unlikelihood training. In *Proceedings of COLING*.

[5] Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xinyu Dai, and Jiajun Chen. 2019. Generating sentences from disentangled syntactic and semantic spaces. In *Proceedings of ACL*.

[6] Gary Bécigneul and Octavian-Eugen Ganea. 2019. Riemannian adaptive optimization methodsRiemannian adaptive optimization methods. In *Proceedings of ICLR*.

[7] Gabor Berend. 2011. Opinion Expression Mining by Exploiting Keyphrase Extraction. In *Proceedings of IJCNLP*.

[8] David M Blei, Thomas L Griffiths, and Michael I Jordan. 2010. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)* 57, 2 (2010), 1–30.

[9] David M Blei, Thomas L Griffiths, Michael I Jordan, Joshua B Tenenbaum, et al. 2003. Hierarchical topic models and the nested Chinese restaurant process.. In *Proceedings of NIPS*.

[10] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3 (2003), 993–1022.

[11] Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. In *Proceedings of NAACL*.

[12] Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of IJCNLP*.

[13] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of CoNLL*.

[14] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[15] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences* 509 (2020), 257–289.

[16] Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural Keyphrase Generation via Reinforcement Learning with Adaptive Rewards. In *Proceedings of ACL*.

[17] Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase Generation with Correlation Constraints. In *Proceedings of EMNLP*.

[18] Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. 2019. An Integrated Approach for Keyphrase Generation via Exploring the Power of Retrieval and Extraction. In *Proceedings of NAACL*.

[19] Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive Hierarchical Decoding for Deep Keyphrase Generation. In *Proceedings of ACL*.

[20] Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019. Title-Guided Encoding for Keyphrase Generation. In *Proceedings of AAAI*.

[21] Ziye Chen, Cheng Ding, Zusheng Zhang, Yanghui Rao, and Haoran Xie. 2021. Tree-structured topic modeling with nonparametric neural variational inference. In *Proceedings of ACL*.

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

[23] Ran Ding, Ramesh Nallapati, and Bing Xiang. 2018. Coherence-aware neural topic modeling. In *Proceedings of EMNLP*.

[24] Minh N Do. 2003. Fast approximation of Kullback-Leibler distance for dependence trees and hidden Markov models. *IEEE signal processing letters* 10, 4 (2003), 115–118.

[25] Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of ACL*.

[26] Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of IJCAI*.

[27] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*.

[28] Sujatha Das Gollapalli, Xiao-Li Li, and Peng Yang. 2017. Incorporating Expert Knowledge into Keyphrase Extraction. In *Proceedings of AAAI*.

[29] Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of WWW*.

[30] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of ACL*.

[31] Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of ACL*.

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

[33] Anette Hulth and Beáta B Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of ACL*.

[34] Masaru Isonuma, Junichiro Mori, Danushka Bollegala, Ichiro Sakata, et al. 2020. Tree-Structured Neural Topic Model. In *Proceedings of ACL*.

[35] Jihyuk Kim, Young-In Song, and Seung-won Hwang. 2021. Web Document Encoding for Structure-Aware Keyphrase Extraction. In *Proceedings of SIGIR*.

[36] Joon Hee Kim, Dongwoo Kim, Suin Kim, and Alice Oh. 2012. Modeling topic hierarchies with the recursive chinese restaurant process. In *Proceedings of CIKM*.

[37] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 Task 5 : Automatic Keyphrase Extraction from Scientific Articles. In *Proceedings of fifth International Workshop on Semantic Evaluation*.

[38] Diederik Kingma and Max Welling. 2014. Efficient gradient-based inference through transformations between bayes nets and neural nets. In *Proceedings of ICML*.

[39] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of ICRL*.

[40] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

[41] Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. *Large dataset for keyphrases extraction*. Technical Report. University of Trento.

[42] Ponnambalam Kumaraswamy. 1980. A generalized probability density function for double-bounded random processes. *Journal of hydrology* 46, 1-2 (1980), 79–88.

[43] Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of EMNLP*.

[44] Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of EMNLP*.

[45] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of EMNLP*.

[46] Olena Medelyan, Eibe Frank, and Ian H Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of EMNLP*.

[47] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep Keyphrase Generation. In *Proceedings of ACL*.

[48] Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. In *Proceedings of ICML*.

[49] Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of ICML*.

[50] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of EMNLP*.

[51] Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of ICADL*.

[52] John Paisley, Chong Wang, David M Blei, and Michael I Jordan. 2014. Nested hierarchical Dirichlet processes. *IEEE transactions on pattern analysis and machine intelligence* 37, 2 (2014), 256–270.

[53] Krutarth Patel and Cornelia Caragea. 2021. Exploiting Position and Contextual Word Embeddings for Keyphrase Extraction from Scientific Papers. In *Proceedings of ECACL*.

[54] Dang Pham and Tuan Le. 2021. Neural Topic Models for Hierarchical Topic Detection and Visualization. In *Proceedings of ECML-PKDD*.

[55] Animesh Prasad and Min-Yen Kan. 2019. Glocal: Incorporating Global Information in Local Convolution for Keyphrase Extraction. In *Proceedings of NAACL*.

[56] Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of AAAI*.

[57] Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. In *Proceedings of ICML*.

[58] Lucas Sterckx, Thomas Demeester, Johannes Deleu, and Chris Develder. 2015. Topical word importance for fast keyphrase extraction. In *Proceedings of WWW*.

[59] Lucas Sterckx, Thomas Demeester, Chris Develder, and Cornelia Caragea. 2016. Supervised keyphrase extraction as positive unlabeled learning. In *Proceedings of EMNLP*.

[60] Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. 2018. Variational recurrent neural machine translation. In *Proceedings of AAAI*.

[61] Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng, and Jian-Yun Nie. 2019. Divgraphpointer: A graph pointer network for extracting diverse keyphrases. In *Proceedings of SIGIR*.

[62] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of NIPS*.

[63] Yixuan Tang, Weilong Huang, Qi Liu, and Beibei Zhang. 2017. QALink: Enriching text documents with relevant Q&A site contents. In *Proceedings of CIKM*.

[64] Nedelina Teneva and Weiwei Cheng. 2017. Salience rank: Efficient keyphrase extraction with topic modeling. In *Proceedings of ACL (Short Papers)*.

[65] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *Proceedings of ICRL*.

[66] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in neural information processing systems* 28 (2015).

[67] Xiaojun Wan and Jianguo Xiao. 2008. Single Document Keyphrase Extraction Using Neighborhood Knowledge.. In *Proceedings of AAAI*.

[68] Chong Wang and David Blei. 2009. Variational inference for the nested Chinese restaurant process. *Advances in Neural Information Processing Systems* 22 (2009), 1990–1998.

[69] Lu Wang and Claire Cardie. 2013. Domain-Independent Abstract Generation for Focused Meeting Summarization. In *Proceedings of ACL*.

[70] Yue Wang, Jing Li, Hou Pong Chan, Irwin King, Michael R. Lyu, and Shuming Shi. 2019. Topic-Aware Neural Keyphrase Generation for Social Media Language. In *Proceedings of ACL*.

[71] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-manning. 1999. KEA: Practical Automatic Keyphrase Extraction. In *Proceedings of JCDL*.

[72] Yumo Xu and Mirella Lapata. 2021. Text Summarization with Latent Queries. *arXiv preprint arXiv:2106.00104* (2021).

[73] Tianchi Yang, Linmei Hu, Chuan Shi, Houye Ji, Xiaoli Li, and Liqiang Nie. 2021. HGAT: Heterogeneous graph attention networks for semi-supervised short text classification. *ACM Transactions on Information Systems (TOIS)* 39, 3 (2021), 1–29.

[74] Hai Ye and Lu Wang. 2018. Semi-Supervised Learning for Neural Keyphrase Generation. In *Proceedings of EMNLP*.

[75] Jiacheng Ye, Ruijian Cai, Tao Gui, and Qi Zhang. 2021. Heterogeneous Graph Neural Networks for Keyphrase Generation. In *Proceedings of EMNLP*.

[76] Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021. ONE2SET: Generating Diverse Keyphrases as a Set. In *Proceedings of ACL*.

[77] Rong Ye, Wenxian Shi, Hao Zhou, Zhongyu Wei, and Lei Li. 2020. Variational template machine for data-to-text generation. In *Proceedings of ICRL*.

[78] Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. One Size Does Not Fit All: Generating and Evaluating Variable Number of Keyphrases. In *Proceedings of ACL*.

[79] Yuxiang Zhang, Yaocheng Chang, Xiaoqing Liu, Sujatha Das Gollapalli, Xiaoli Li, and Chunjing Xiao. 2017. Mike: keyphrase extraction by integrating multidimensional information. In *Proceedings of CIKM*.

[80] Yong Zhang, Yang Fang, and Xiao Weidong. 2017. Deep keyphrase generation with a convolutional sequence to sequence model. In *Proceedings of ICSAI*.

[81] Jing Zhao and Yuxiang Zhang. 2019. Incorporating Linguistic Constraints into Keyphrase Generation. In *Proceedings of ACL*.

[82] Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Proceedings of ACL*.

[83] Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achanauparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical keyphrase extraction from twitter. In *Proceedings of ACL*.