# Towards Automatic Job Description Generation With Capability-Aware Neural Networks

Chuan Qin, *Member, IEEE*, Kaichun Yao, Hengshu Zhu, *Senior Member, IEEE*, Tong Xu, *Member, IEEE*, Dazhong Shen, Enhong Chen, *Senior Member, IEEE*, and Hui Xiong, *Fellow, IEEE*

**Abstract**—A job description shows the responsibilities of the job position and the skill requirements for the job. An effective job description will help employers to identify the right talents for the job, and give a clear understanding to candidates of what their duties and qualifications for a particular position would be. However, due to the variation in experiences, it is always a challenge for both hiring managers and recruiters to decide what capabilities the job requires and prioritize them accordingly on the job description. Also, tedious and expensive human efforts are usually required to prepare a job description. Therefore, in this paper, we investigate how to automate the process to generate job descriptions with less human intervention. To this end, we propose an end-to-end capability-aware neural job description generation framework, namely Cajon, to facilitate the writing of job description. Specifically, we first propose a novel capability-aware neural topic model to distill the various capability information from the larger-scale recruitment data. Also, an encoder-decoder recurrent neural network is designed for enabling the job description generation. In particular, the capability-aware attention mechanism and copy mechanism are proposed to guide the generation process to ensure the generated job descriptions can comprehensively cover relevant and representative capability requirements for the job. Moreover, we propose a capability-aware policy gradient training algorithm to further enhance the rationality of the generated job description. Finally, extensive experiments on real-world recruitment data clearly show our Cajon framework can help to generate more effective job descriptions in an interpretable way. In particular, our Cajon framework has been deployed in Baidu as an intelligent tool for talent recruitment.

**Index Terms**—Job description generation, recruitment analysis, topic model

## 1 Introduction

A successful hiring process always starts with a carefully written job description. As a diverse tool of talent recruitment, job description details the primary functions of the job, how the tasks will be carried out, and the necessary capabilities needed to perform the job. An effective job description will help employers to identify the right talents for the job, and give a clear understanding to candidates of what their duties and qualifications for a particular position would be. Beyond the hiring process, it can also serve as a way for employees to measure job performance and give managers a framework for evaluation. In contrast, a poorly-crafted or misleading job description will result in various management problems, including person-job mismatch, workplace confusion, miscommunication and even legal risks.

While the significance of job description has been widely recognized, according to a recent research report from Allegis Group [1], many organizations are facing difficulties when writing job descriptions, since only 39% of candidates think the organizations' job descriptions are clear and easy to understand. Indeed, it is far from a trivial task to write an effective job description, especially for those requiring complicated job qualifications. In real-world practice, a long-standing challenge for both hiring managers and recruiters is to consider what capabilities the job requires and prioritize them accordingly on the job description, and meanwhile, translate the subtleties of job descriptions to the written word [2]. In the past decade, recruitment data analysis has attracted wide research attentions in the field of data mining. For example, researchers have studied the problem of recruitment market trend modeling [3], [4], person-job fit in talent recruitment [5], [6], and the job skill popularity modeling [7], [8]. While these studies can provide some general market-driven guidance for improving job descriptions, there still exists a crucial demand of building an intelligent system for facilitating the writing of job descriptions.

To this end, in this paper we introduce a novel end-to-end approach for automatic job description generation, with a focus on the complicated capability requirements. Along this line, there are two critical tasks. The first one targets at "what to write", that is, what are the fine-grained capability requirements for each job posting. Indeed, companies usually post a

- *Chuan Qin, Kaichun Yao, and Hengshu Zhu are with Talent Intelligence Center, Baidu Inc., Beijing 100085, China. E-mail: {chuanqin0426, zhuhengshu}@gmail.com, yaokaichun@outlook.com.*
- *Tong Xu, Dazhong Shen, and Enhong Chen are with the School of Computer Science, University of Science and Technology of China, Hefei 230027, China. E-mail: {tongxu, cheneh}@ustc.edu.cn, sdz@mail.ustc.edu.cn.*
- *Hui Xiong is with Artificial Intelligence Thrust, Hong Kong University of Science and Technology, Guangzhou 511458, China. E-mail: xionghui@gmail.com.*

*Manuscript received 3 July 2020; revised 22 Sept. 2021; accepted 12 Jan. 2022. Date of publication 25 Jan. 2022; date of current version 3 Apr. 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 61836013, 91746301, and 62072423. (Corresponding authors: Hengshu Zhu and Hui Xiong.) Recommended for acceptance by Y. Chen. Digital Object Identifier no. 10.1109/TKDE.2022.3145396*

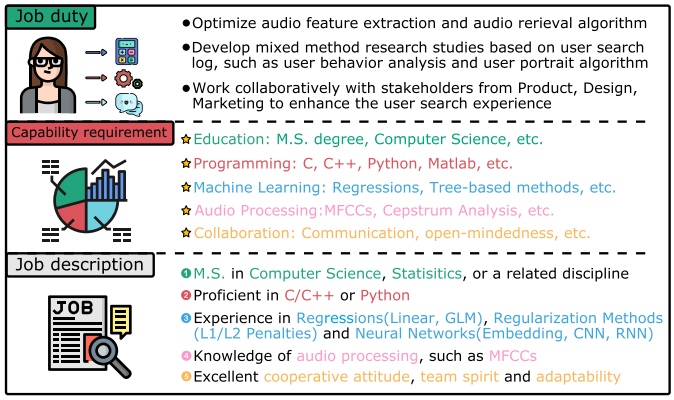



Fig. 1. A motivating example of writing job description.

job opening when they need to enroll talents to support some specific business, or fill the role of some resigned employees. In other words, hiring managers and recruiters usually know the clear responsibilities of each job position, i.e., job duties, before writing the job description. Fig. 1 shows a toy job description for recruiting an algorithm engineer, the job duty of which includes optimizing audio search algorithms, analyzing user behavior data, and further improving the user search experience. Experienced recruiters could identify the necessary capabilities required by this job, including *education*, *programming*, *machine learning*, *audio processing* and *collaboration* respectively. Therefore, the major challenge is how to catch the latent relationships between job duty and the capability requirement. Furthermore, the second task targets at "how to write", that is, how to generate the job description according to the capability requirements and translate them to the written word. Intuitively, we can leverage abundant recruitment data to train the natural language generation model, such as sequence-to-sequence model and its variants [9], [10], [11], [12]. However, prior arts pay more attention to the smoothness of the generated results than the rationality of the generated results. Therefore, a challenge that cannot be neglected is how to design a capability-aware natural language generation model to ensure the effectiveness of the generated job descriptions.

To be specific, here we propose a **c**apability-**a**ware neural **j**ob descripti**on** generation framework, namely Cajon (a famous instrument originally from Peru, which is also the internal code name of corresponding product in Baidu), to facilitate the writing of job description. We first propose a novel capability-aware neural topic model to distill the various capability information from the real-world recruitment data. Second, an encoder-decoder recurrent neural network is designed for enabling the job description generation. In particular, the capability-aware attention mechanism and copy mechanism are proposed to guide the generation process to ensure the generated job descriptions comprehensively cover relevant and representative capability requirements for job postings. Third, we propose a capability-aware policy gradient training algorithm to directly optimize the non-differentiable evaluation measures, especially the rationality of the generated skills, which can further enhance the rationality of the generated job description. Finally, extensive experiments conducted on real-world recruitment dataset clearly validate the effectiveness of our Cajon framework and its variants compared with state-of-the-art baselines. Further case studies and discussions also demonstrate the interpretability of our model for learning meaningful capability topics. The Cajon framework has been deployed in Baidu as an intelligent tool for enhancing talent recruitment.

## 2 Related Work

Generally, the related works of this paper can be grouped into three areas, namely Recruitment Analysis, Natural Language Generation and Topic Model. We will introduce them in turn below.

*Recruitment Analysis.* Recruitment has always been a critical part to support the success of the business, which has attracted many researchers to conduct recruitment analysis from different perspectives, such as person-job fit [5], [6], [13], talent career path analysis [14], [15] and job interview assessment [16], [17]. Among them, some studies focus on leveraging the accumulation of recruitment data to analyze capability requirements in the recruitment market. *Schlee et al.* analyzed the difference in capability distribution between different levels of marketing posting [4]. *Wade et al.* further explored the mix of organizational and technical skills demanded for some particular job position and their impact on job performance [18]. The above traditional methods are mainly based on the knowledge of domain experts and classic statistical methods. In recent years, several efforts have been made in novel perspectives by exploiting data mining techniques. For instance, *Zhu et al.* proposed an unsupervised sequential latent variable model for forecasting the evolutionary trend of demand for talents with different capabilities in the recruitment market [3]. *Xu et al.* further measured the popularity of job skills in the recruitment market by using a novel probabilistic topic model [8]. Moreover, *Wu et al.* proposed a tensor factorization based approach to achieve job skill demand analysis [7].

However, most of the previous researches focused on the analysis of competence requirements for the recruitment market, so it is difficult to efficiently and intuitively assist human resources in customizing job descriptions of different job postings. Recently, *Liu et al.* introduced a data-driven two-step neural model, namly SAMA, to write job posting [19]. However, their work depends on several side information, such as the company scale. It makes their solutions less robust, because such information often appears vacant and noisy, which may affects the effectiveness of the model. Differently, in this paper, we propose a more general end-to-end method, namely Cajon, which includes a novel neural topic model to make the generated job descriptions have good interpretable at capability level. And, we propose the capability-aware attention and copy mechanisms to make the generated results more smooth and reasonable. In addition, we introduce a capability-aware policy gradient training algorithm to directly optimize the non-differentiable evaluation measures, especially the rationality of the generated skills. It is more concise and effective than SAMA, which used an additional skill prediction decoder.

*Natural Language Generation.* Natural Language Generation (NLG) targets at transforming the structured data into natural language, covering a wide range of natural language process tasks, such as machine translation [9], text summarization [20]

and chatbot [21], [22]. Traditionally, prior arts usually adopted template-based [23], [24] and rule-based [25] methods. During the last decade, with the development of deep learning techniques, a series of neural-based models have shown promising results in NLG tasks. Among them, sequence-to-sequence model becomes a common choice for text generation, which has been first proposed by *Sutskever et al.* [9]. Subsequently, the researchers introduced a series of variants to further improve the performance. For instance, *Bahdanau et al.* extended the sequence-to-sequence model by introducing an attention mechanism to align words in the source and target sentences [10]. *Xia et al.* proposed a deliberation network that contains a two-pass decoder to improve the effectiveness of neural machine translation [26]. *Tu et al.* further proposed a coverage attention mechanism to address the over-generation problem in text generation [11]. *Gu et al.* introduced a copy mechanism to handle the rare word generation problem by copying the source words to the positions of the not-found target words [12]. Moreover, *Dong et al.* demonstrated the utility of pre-trained language model in text generation tasks by combining unidirectional, bidirectional and sequence-to-sequence unsupervised language modeling objectives. [27].

Recently, reinforcement learning has been applied in several Natural Language Generation tasks, which can further improve the generation performance by optimizing those discrete and non-differentiable evaluation measures during the training processing. For instance, *Ranzato et al.* trained the sequence-to-sequence model by employing the policy gradient method [28] to directly optimize ROUGE [29] and BLEU [30] for three NLG tasks including text summarization, machine translation and image captioning [31]. *Li et al.* optimized several handcrafted rewards, such as ease of answering, information flow and semantic coherence with policy gradient method for obtaining diverse, coherent responses in dialogue generation [32]. *Rennie et al.* further introduced the self-critical policy gradient training algorithm to solve the high variance problem encountered by the above method when calculating the expected gradient [33]. Such method and its variants have been demonstrated its effectiveness in image caption [33], text summarization [34] and reading comprehension [35]. Furthermore, actor-critic method can also reduce the variance of the model. Specifically, *Bahdanau et al.* presented an actor-critic method with Q-learning for the machine translation [36], and *Chen et al.* applied a advantage actor-critic method with sentence-level metric rewards for the text summarization [37].

In this paper, we follow some outstanding ideas of the above studies according to the properties of customizing job description generation, and propose an end-to-end CANJDG framework based on the sequence-to-sequence model with capability-aware attention copy mechanisms. Moreover, the proposed capability-aware policy gradient training algorithm can further enhance the rationality of the capabilities covered in the generated job description.

*Topic Model.* Probabilistic topic model is another closely related area of our approach, which targets at exploiting human interpretable latent topics from coherent words in textual data. As one of the most famous probabilistic topic models, Latent Dirichlet Allocation (LDA) [38] which based on Bayesian inference has a number of variants [39], [40], [41]. Among them, some researches focus on exploiting latent topics from short texts and have a series of practical applications. For instance, *Yan et al.* proposed a Biterm Topic Model, which explicitly modeled the co-occurrence patterns in the whole short text corpus, and proofed that this model outperforms LDA on short texts [42]. In order to model a special type of short text on social media, such as Twitter, *Zhao et al.* designed a twitter-LDA model based on the assumption that each tweet usually originates from a single topic [41]. Besides, *Lin et al.* proposed a dual-sparse topic model to address the sparsity of the topic representation for documents and the word representation for topics when analyzing user-generated short text on social media.

In the past years, topic model has been already successfully applied in various research fields, such as recommender system [43], [44] and social media [45], [46]. Moreover, it has been further extended to explore extra content information, such as diverse text corpora [39], document labels [40], [47] and authorship [48]. However, these models rely on carefully customized graphical models and tailored inference algorithms. Recently, with the development of deep learning techniques, Neural Topic Model (NTM) was proposed to explore latent topics by employing neural variational inference to train topic model [49], [50]. Moreover, NTM has good versatility and scalability, because it can facilitate to end-to-end training with other neural network and be exploited in various application scenarios without carefully customized graphical models and inference algorithms, such as text classification [51] and keywords extraction [52].

In addition, because the concept of the topic in the probabilistic topic model can correspond to the human notion of it, topic model is also used to improve the interpretability of the model in different fields. For instance, *Ramage et al.* introduced Labeled-LDA to handle the credit attribution problem in document tagging using an interpretable way by defining a one-to-one correspondence between LDA's latent topic and tags [40]. *McAuley et al.* proposed a product recommendation model which aligns hidden factors in product ratings with hidden topics in product review, so that it can explain user-item compatibility by examining the topics corresponding to matching components of their latent factors [53]. *Shen et al.* designed a joint learning model on interview assessment [54]. In particular, they leveraged the neural topic model to learn the different semantic spaces of the job description, resume and interview assessment data, which can learn the relationship of skill topics in different kinds of data, thereby interpretively recommend appropriate assessment skills to interviewers.

Different from prior arts, in this paper, we design a novel capability-aware neural topic model to distill the various capability information from the larger-scale recruitment data and guide the job description generation process, which improves the effectiveness and interpretability of our model.

## 3 The Cajon Model

In this section, we will introduce our Cajon framework. We will first give a formal definition of the job description generation problem (Section 3.1). Then, we will introduce our Cajon framework in detail. Fig. 2 shows the overall architecture, which mainly consists of two components, namely capability-aware neural topic model (Section 3.2) and capability-aware neural job description generation model

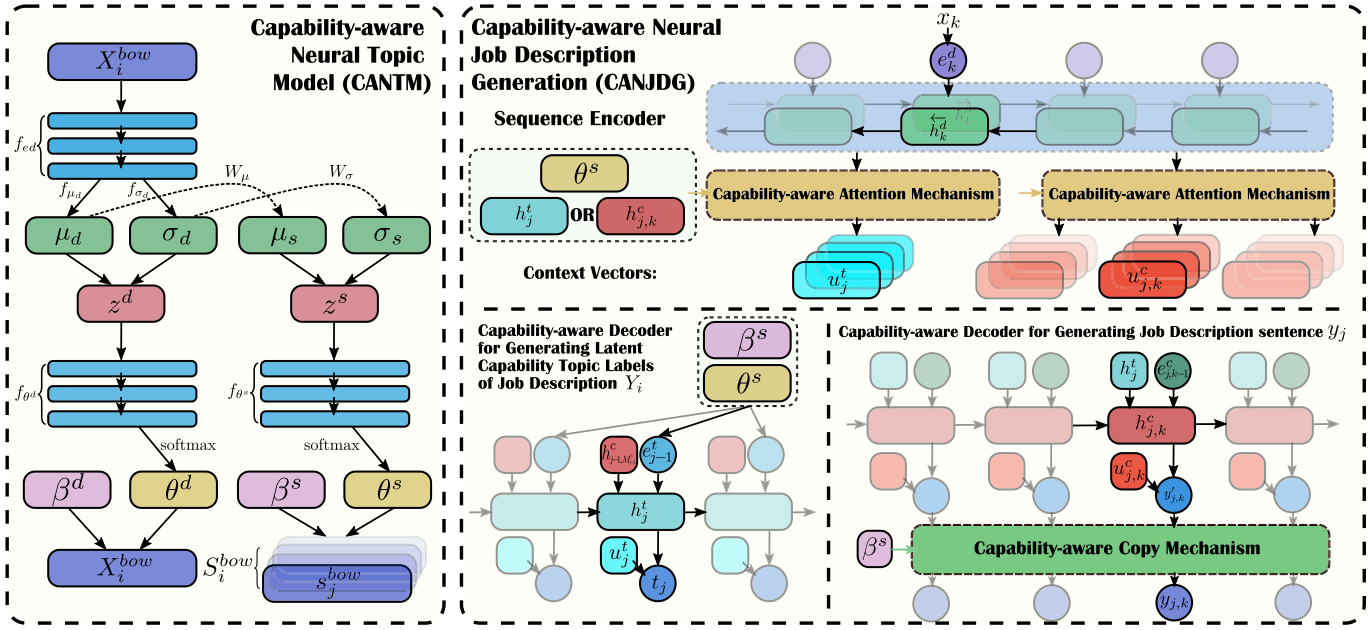



Fig. 2. An illustration of the proposed Cajon framework without capability-aware policy gradient training processing, which can be separated into two components, namely capability-aware neural topic model and capability-aware neural job description generation model.

(Section 3.3). Finally, we will describe how to jointly train the above models with using a capability-aware policy gradient training algorithm(Section 3.4). For facilitating illustration, we list some important mathematical notations used throughout this paper in Table 1.

TABLE 1
Mathematical Notations

| Symbol | Description |
|---|---|
| $x_j$ | The $j$th word of job duty $X_i$; |
| $y_j$ | The $j$th sentence of job description $Y_i$; |
| $y_{j,k}$ | The $k$th word of $j$th sentence at job description $Y_i$; |
| $s_{j,k}$ | The $k$th capability word of $j$th sentence at job description $Y_i$; |
| $X_i^{bow}$ | The bag-of-words input of job duty $X_i$; |
| $s_j^{bow}$ | The bag-of-words input of the capability word list $s_j$ at $j$th sentence of job description $Y_i$; |
| $\beta_j^d$ | The word distribution for $j$th latent topic $k_j^d$ in job duty; |
| $\beta_j^s$ | The word distribution for $j$th latent topic $k_j^s$ in job description; |
| $\theta^d$ | The topic mixtures of job duty $X_i$; |
| $\theta^s$ | The topic mixtures of capability word lists $S_i$ in job description $Y_i$; |
| $t_j$ | The latent capability topic label of $j$th job description sentence $y_j$; |
| $h_k^d$ | The hidden state of the $k$th word $x_k$ in the sequence encoder; |
| $h_j^t$ | The hidden state of the $j$th latent capability topic label $t_j$ in the capability-aware sequence decoder; |
| $h_{j,k}^c$ | The hidden state of the $k$th word of the $j$th job description sentence $y_j$ in the capability-aware sequence decoder; |
| $M^d$ | The word count of job duty $X_i$; |
| $M_j^c$ | The word count of the $j$th job description sentence $y_j$; |
| $M_j^s$ | The capability word count of the $j$th job description sentence $y_j$; |
| $N$ | The sentence count of job description $Y_i$. |

### 3.1 Problem Definition

As we mentioned before, in this paper, we target at automatic job description generation, with a focus on the complicated capability requirements. Formally, given a collection of recruitment documents $\mathcal{C}$ with $|\mathcal{C}|$ unique job postings, i.e., $\mathcal{C} = \{C_i = (X_i, Y_i)\}_{i=1}^{|\mathcal{C}|}$, where $X_i$ is the job duty which describes the responsibilities for the $i$th job posting and $Y_i$ is the job description which describes the capability requirements of this job. Specifically, for each job duty $X_i$, it is assumed to contain $M^d$ words, i.e., $X_i = \{x_1, x_2, \ldots, x_{M^d}\}$. Differently, since a job description often includes multiple sentences to introduce different capability requirements, we denote each job description $Y_i$ as $Y_i = \{y_1, y_2, \ldots, y_N\}$, where $y_j$ is the $j$th sentence in $Y_i$. For instance, there are 5 job description sentences in Fig. 1, i.e., $N = 5$, which introduce the capability requirements of *education*, *programming*, *machine learning*, *audio processing* and *collaboration* respectively. Moreover, each $y_j$ is assumed to contain $M_j^c$ words, i.e., $y_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,M_j^c}\}$.

In addition, in order to analyze the fine-grained capability requirements for each job posting, here we follow the idea in [17] to train a neural model to extract the capability words in each sentence $y_j$ in job description $Y_i$. With the further manual annotation of the extracted capability words, we can generate the corresponding capability word list for $y_j$, i.e., $s_j = \{s_{j,1}, s_{j,2}, \ldots, s_{j,M_j^s}\}$. And, we denote all capability word lists of $Y_i$ as $S_i = \{s_1, s_2, \ldots, s_N\}$. Along this line, we define our job description generation problem as follows, which has a focus on the complicated capability requirements generation:

**Definition 3.1 (Problem Definition).** *Given a set of recruitment documents $\mathcal{C}$, where each $C_i \in \mathcal{C}$ contains a job duty $X_i$ and a job description $Y_i$. The target of job description generation is to learn a model $\mathcal{M}$ which can generate fluent and rational job description $Y_{new}$ when a new job duty $X_{new}$ is given.*

### 3.2 Capability-Aware Neural Topic Model (CANTM)

As we mentioned before, to handle the job description generation problem, we first need to know the corresponding

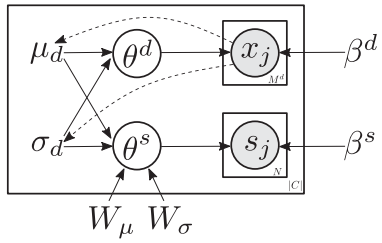



Fig. 3. The illustration of CANTM as a directed graph. We use solid lines to denote the generative model and dashed lines to denote the variational approximation.

capability requirements for this job posting. Here, inspired by the idea from *Miao et al.* [50], we propose a novel Capability-aware Neural Topic Model (CANTM) to exploit the latent capability topics for job duty and job description. The graphical model of CANTM is shown in Fig. 3. In the following, we introduce the *generation* and *inference* processes of CANTM, respectively.

*CANTM Generation.* To model the latent semantics in job duty and job description, we assume there exists a finite number of latent topics $K^d$ and $K^s$ over job duty and job description, respectively. Each topic $k_j^d$ and $k_j^s$ are represented by the word distribution $\beta_j^d$ and $\beta_j^s$

$$\beta_j^d = \text{softmax}(v_d \cdot (t_d)_j^\top), \quad \beta_j^s = \text{softmax}(v_s \cdot (t_s)_j^\top), \tag{1}$$

where $t_d \in \mathbb{R}^{K^d \times H}$ and $t_s \in \mathbb{R}^{K^s \times H'}$ are topic-based parameters; and $v_d \in \mathbb{R}^{V^d \times H}$ and $v_s \in \mathbb{R}^{V^s \times H'}$ are word-based parameters, will be learned during the training processing. Here, $V^d$ and $V^s$ are the vocabulary size of job duty and job description, respectively. In particular, we only consider the capability word lists $S_i$ as the data input for the job description part in CANTM, which can reduce the input noise and enhance the performance for learning the latent capability topics in the job description.

Then, similar to LDA-style topic models, we assume each job duty $X_i$ and capability word lists $S_i$ in job description $Y_i$ have the topic mixtures $\theta^d$ and $\theta^s$, respectively, $\theta^d \in \mathbb{R}^{K^d}$ and $\theta^s \in \mathbb{R}^{K^s}$. Here, following the idea from *Miao et al.* [50], the topic mixtures $\theta^d$ and $\theta^s$ are generated via Gaussian softmax construction. Specifically, the generative process for job duty $X_i$ is:

- Draw latent variable $z^d \sim \mathcal{N}(\mu_d, \sigma_d^2)$
- $\theta^d = \text{softmax}(f_{\theta^d}(z^d))$
- For the $l$th word in $X_i$:
  - Draw word $x_l \sim \theta^d \cdot \beta^d$,

where $\mu_d$ and $\sigma_d$ are the prior parameters, and $f_{\theta^d}(\cdot)$ is a neural perception actived by a non-linear transfromation following [51].

Differently, for the generative process of the capability word lists $S_i$ in job description $Y_i$, we involve a natural assumption based on the observation in dataset, i.e., each capability word list $s_j$ on sentence $y_j$ of $Y_i$ is usually about a single capability topic. Along this line, the generative process is:

- Draw latent variable $z^s \sim \mathcal{N}(\mu_s, \sigma_s^2)$
- $\theta^s = \text{softmax}(f_{\theta^s}(z^s))$
- For the capability word list $s_j$ on $j$th sentence $y_j$ of $Y_i$:
  - Draw $s_j = \{s_{j,k}\}_{k=1}^{M_j^s}$ with probability:

$$p(s_j|\theta^s, \beta^s) = \theta^s \cdot \prod_{k=1}^{M_j^s} \beta^s_{*,s_{j,k}},$$

where $\mu_s$ and $\sigma_s$ are the prior parameters and $\beta^s_{*,s_{j,k}}$ denotes the column vector for capability word $s_{j,k}$ in $\beta^s$. Here, $\prod_{k=1}^{M_j^s} \beta^s_{l,s_{j,k}}$ calculates the probability of $s_j$ contains skills $\{s_{j,k}\}_{k=1}^{M_j^s}$ corresponding to the topic $k_l^s$.

Moreover, to model the strong correlation between the each job duty $X_i$ and the capability word lists $S_i$ in job description $Y_i$, we assume that the prior parameters of their latent topics have the following mapping relationship:

$$\mu_s = W_\mu \mu_d, \quad \log \sigma_s = W_\sigma (\log \sigma_d), \tag{2}$$

where the $W_\mu$ and $W_\sigma$ are the trainable parameters.

*CANTM Inference.* According to the generation process of our CANTM, the marginal likelihood is

$$\begin{aligned} &p(X_i, S_i|\mu_d, \mu_s, \sigma_d, \sigma_s, \beta^d, \beta^s) \\ &= \int p(\theta^d|\mu_d, \sigma_d^2) \prod_{k=1}^{M^d} p(x_k|\theta^d, \beta^d) \mathrm{d}\theta^d \\ &\quad \cdot \int p(\theta^s|\mu_s, \sigma_s^2) \prod_{j=1}^{N} p(s_j|\theta^s, \beta^s) \mathrm{d}\theta^s. \end{aligned} \tag{3}$$

Here, we use the neural variational inference [50] to approximate posterior distributions over $\theta^d$ and $\theta^s$. The variational low bound for the log-likelihood according to Equation (3) is

$$\begin{aligned} \mathcal{L} = &\mathbb{E}_{q(\theta^d)} \left[ \sum_{k=1}^{M^d} \log p(x_k|\theta^d, \beta^d) \right] - D_{KL}(q(\theta^d)||p(\theta^d|\mu_d, \sigma_d)) \\ &+ \mathbb{E}_{q(\theta^s)} \left[ \sum_{j=1}^{N} \log p(s_j|\theta^s, \beta^s) \right] - D_{KL}(q(\theta^s)||p(\theta^s|\mu_s, \sigma_s)), \end{aligned} \tag{4}$$

where $q(\theta^d)$ and $q(\theta^s)$ are the variational distribution approximating the true posterior $p(\theta^d|X_i, S_i)$ and $p(\theta^s|X_i, S_i)$. And $D_{KL}$ denotes the Kullback-Leibler divergence. The detailed description of Equation (4) can be found in Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2022.3145396.

To generate the variational parameters $\mu_d(X_i, S_i)$, $\log\sigma_d(X_i, S_i)$, $\mu_s(X_i, S_i)$ and $\log\sigma_s(X_i, S_i)$, we follow the idea in [55] to estimate $\mu_d$, $\sigma_d$, $\mu_s$ and $\sigma_s$ only from the input $X_i$. It allows us to use the CANTM model to explore the latent capability-aware topic mixture $\theta_d$ and $\theta_s$ only by the job duty. So, here we introduce an inference network conditioned on the observed job duty $X_i$ and combine Equation (2) to generate the above variational parameters

$$\begin{aligned} \mu_d(X_i, S_i) &= f_{\mu_d}(f_{e_d}(X_i^{bow})), \\ \mu_s(X_i, S_i) &= W_\mu \mu_d(X_i, S_i), \\ \log\, \sigma_d(X_i, S_i) &= f_{\sigma_d}(f_{e_d}(X_i^{bow})), \\ \log\, \sigma_s(X_i, S_i) &= W_\sigma \log\, \sigma_d(X_i, S_i), \end{aligned} \tag{5}$$

where $X_i^{bow}$ denotes the bag-of-words vector of the input data $X_i$, $f_{e_d}(\cdot)$ is a neural perception which is actived by a non-linear transfromation and $f_{\mu_d}(\cdot)$, $f_{\sigma_d}(\cdot)$ are the linear neural perception.

Along this line, we can directly minimize the following loss function of CANTM for each instance pair $(X_i, S_i)$ during the training process, that is

$$\mathcal{L}_{CANTM} = -\sum_{k=1}^{M^d} \log(\theta^d \cdot \beta^d_{*,x_k}) + D_{KL}(q(\theta^d)||p(\theta^d)) - \sum_{j=1}^{N} \log\left(\theta^s \cdot \prod_{k=1}^{M^s_j} \beta^s_{*,s_{j,k}}\right) + D_{KL}(q(\theta^s)||p(\theta^s)). \quad (6)$$

Therefore, we can infer all the parameters in CANTM and further exploit the latent capability topics for each job.

### 3.3 Capability-Aware Neural Job Description Generation (CANJDG)

After learning the latent capability topics by CANTM, we turn to introduce how to generate job description with an encoder-decoder neural model. Specifically, as shown in Fig. 2, it mainly contains two components, including a *sequence encoder* which extracts the semantic information from an input job duty document $X_i$, and a *capability-aware sequence decoder* which generates each word in job description $Y_i$ by guiding from the latent capability topics.

*Sequence Encoder.* Here, we first use a look-up layer to transform each word $x_k$ in job duty document $X_i$ into an embedding vector $e^d_k$. Then we employ a bidirectional Long Short-Term Memory network (LSTM) to encode the input sequence $\{e^d_1, e^d_2, \ldots, e^d_{M^d}\}$. Specifically, we have

$$\overrightarrow{h^d_k} = LSTM(e^d_k, \overrightarrow{h^d_{k-1}}), \quad \overleftarrow{h^d_k} = LSTM(e^d_k, \overleftarrow{h^d_{k+1}}), \quad (7)$$

where $e^d_k = W_{e^d} x_k$ is the word embedding of the $k$th word $x_k$. Finally, we use the $h^d_k = [\overrightarrow{h^d_k}; \overleftarrow{h^d_k}]$ to denote the hidden state of the $x_k$ in the sequence encoder.

*Capability-Aware Sequence Decoder.* Now, we turn to construct a decoder to generate each word in job description $Y_i$. Generally, during the generative process, we first estimate the capability topic $t_j$ for each sentence $y_j$ in $Y_i$, and then predict each word $y_{j,k}$ with the following probability:

$$p(y_{j,k}|X_i) = p(y_{j,k}|y_{<j}, y_{j,<k}, \mathcal{H}, \theta^s, t_j), \quad (8)$$

where $y_{<j}$ denotes the sentences $\{y_1, y_2, \ldots, y_{j-1}\}$, $y_{j,<k}$ denotes the words $\{y_{j,1}, y_{j,2}, \ldots, y_{j,k-1}\}$, and $\mathcal{H} = \{h^d_1, h^d_2, \ldots, h^d_{M^d}\}$ denotes all the hidden states in sequence encoder. $\theta^s$ is the latent capability topic mixture of $Y_i$, which is learned from CANTM, and $t_j \in [1, K^s]$ is the capability topic label of sentence $y_j$.

Specifically, our capability-aware sequence decoder is built upon two unidirectional LSTMs. Let $h^t_j$ and $h^c_{j,k}$ denote the hidden states in the LSTMs for generating the $j$th capability topic $t_j$ and $k$th word $y_{j,k}$ in sentence $y_j$, which are computed via

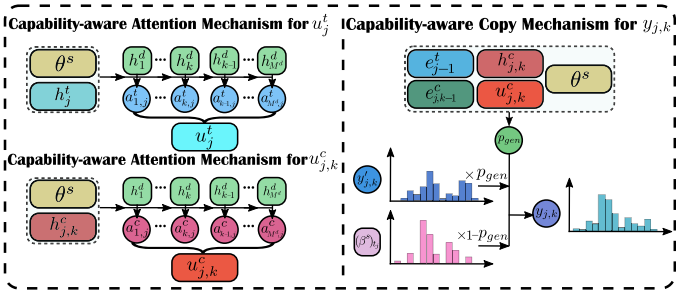



Fig. 4. An illustration of the proposed capability-aware attention mechanism and capability-aware copy mechanism.

$$h^t_j = LSTM([e^t_{j-1}; \theta^s; h^c_{j-1,M^c_{j-1}}], h^t_{j-1}),$$
$$h^c_{j,k} = LSTM([e^c_{j,k-1}; \theta^s; h^t_j], h^c_{j,k-1}),$$
$$e^t_j = W_{e^t} t_j, \quad e^c_{j,k} = W_{e^c} y_{j,k}, \quad (9)$$

where $e^t_j$, $e^c_{j,k}$ are the embedding vectors of capability topic label $t_j$ and word $y_{j,k}$, respectively, $W_{e^t}$ and $W_{e^c}$ are trainable parameters. Here we use $M^c_j$ to denote the length of sentence $y_j$. The first time step of hidden states $h^t_0$ and $h^c_{1,0}$ are initialized by the encoding vector $h^d_{M^d}$, and the $h^c_{j,0}$ is initialized by the $h^c_{j-1,M^c_{j-1}}$.

Additionally, we use two capability-aware attention mechanisms to capture the important context feature from the encoder states $\mathcal{H}$ for enhancing the performance of generative process, where are illustrated on the left side of Fig. 4. Here we follow the idea of [10], and further consider the latent capability topic mixture $\theta^s$ to calculate the attention score on $h^d_l$ when predicting the $j$th latent capability label $t_j$ and $k$th word $y_{j,k}$ in $y_j$, i.e.,

$$\alpha^t_{l,j} = \frac{\exp(g^t_{l,j})}{\sum_{p=1}^{N} \exp(g^t_{p,j})}, \quad \alpha^c_{l,j,k} = \frac{\exp(g^c_{l,j,k})}{\sum_{p=1}^{M^c_j} \exp(g^c_{p,j,k})},$$
$$g^t_{l,j} = v^\top_{\alpha^t} \tanh(W_{\alpha^t}[h^d_l; h^t_j; \theta^s] + b_{\alpha^t}),$$
$$g^c_{l,j,k} = v^\top_{\alpha^c} \tanh(W_{\alpha^c}[h^d_l; h^c_{j,k}; \theta^s] + b_{\alpha^c}), \quad (10)$$

where $v_{\alpha^t}$, $v_{\alpha^c}$, $W_{\alpha^t}$, $W_{\alpha^c}$, $b_{\alpha^t}$ and $b_{\alpha^c}$ are the trainable parameters. Then we calculate the capability-aware context vectors $u^t_j$ and $u^c_{j,k}$ by:

$$u^t_j = \sum_{l=1}^{N} \alpha^t_{l,j} h^d_l, \quad u^c_{j,k} = \sum_{l=1}^{M^c_j} \alpha^c_{l,j,k} h^d_l. \quad (11)$$

Now, we can predict the $j$th latent capability label $t_j$ and $k$th word $y_{j,k}$ in $y_j$ according to

$$p(t_j|t_{<j}, \mathcal{H}, \theta^s) = \text{softmax}(W_t[h^t_j; u^t_j; \theta^s] + b_t), \quad (12)$$

$$p(y_{j,k}|y_{<j}, y_{j,<k}, \mathcal{H}, \theta^s, t_j) = \text{softmax}(W_c[h^c_{j,k}; u^c_{j,k}; \theta^s] + b_c), \quad (13)$$

where $W_t$, $W_c$, $b_t$ and $b_c$ are the trainable parameters.

Moreover, we adopt a capability-aware copy mechanism, which allows our decoder to directly copy the words from the capability vocabulary, where are illustrated on the right side of Fig. 4. Specifically, we define a generation probability $p_{gen} \in [0, 1]$ when generating $k$th word $y_{j,k}$ in $y_j$ as

$$p_{gen} = \text{sigmoid}(W_p[e^t_j; e^c_{j,k-1}; h^c_{j,k}; u^c_{j,k}; \theta_s] + b_p), \quad (14)$$

with trainable parameters $W_p$ and $b_p$. Then we update the Equation (13) to obtain the following probability distribution according to the capability vocabulary

$$p(y_{j,k}|y_{<j}, y_{j,<k}, \mathcal{H}, \theta^s, t_j) = p_{gen}\text{softmax}(W_c[h^c_{j,k}; u^c_{j,k}; \theta^s] + b_c) + (1 - p_{gen})(\beta^s)_{t_j}, \quad (15)$$

where $(\beta^s)_{t_j}$ is the word distribution for latent capability topic $t_j$ learned from our CANTM model.

Finally, in our neural generation model, we minimize the following cross entropy loss functions for latent capability topic labels and generated job description for each instance pair $(X_i, Y_i)$

$$\mathcal{L}_{CTL} = -\sum_{j=1}^{N} \log p(t_j|X_i, \theta_s),$$
$$\mathcal{L}_{GJD} = -\sum_{j=1}^{N}\sum_{k=1}^{M^s_j} \log p(y_{j,k}|X_i, \theta_s). \quad (16)$$

### 3.4 Capability-Aware Policy Gradient Training Algorithm (CAPGTA)

Before introducing our reinforcement learning-based training algorithm, namely Capability-aware Policy Gradient Training Algorithm (CAPGTA), we will first show a basic end-to-end training process to jointly learn the all the parameters in the above Capability-aware Neural Topic Model (CANTM) and Capability-aware Neural Job Description Generation (CANJDG).

Specifically, since CANTM is based on neural variational inference, it allows us jointly optimize the composite loss function with the consideration of $\mathcal{L}_{CANTM}$, $\mathcal{L}_{CTL}$ and $\mathcal{L}_{GJD}$ as following:

$$\mathcal{L}^* = \mathcal{L}_{GJD} + \lambda_1\mathcal{L}_{CANTM} + \lambda_2\mathcal{L}_{CTL}, \quad (17)$$

where $\lambda_1$ and $\lambda_2$ are the hyperparameters to balance the effects of different models. Specially, we employ the teacher forcing algorithm [56] upon the CAPGTA model during the training process. Here, when calculating hidden state $h^t_j$ and $h^c_{j,k}$ at each sequence decoding time step, we take the previous word $y_{j,k-1}$ from ground-truth during the training process and from the predicted word in test. For the latent capability topic label $t_{j-1}$, according to the generative process in CANTM, we use

$$t_{j-1} = \arg\max_l \left\{ (\theta^s)_l \left( \prod_{k=1}^{M^s_{j-1}} \beta^s_{*,s_{j-1,k}} \right)_l \right\}_{l=1}^{K^s}, \quad (18)$$

in training processing, and use the predicted capability topic label in test.

However, minimizing $\mathcal{L}^*$ does not always generate the best job descriptions, since it does not directly optimize discrete evaluation metrics, such as ROUGE [29] and BLEU [30]. Moreover, we hope to be able to more intuitively optimize the accuracy of the capabilities involved in the generated job description, which can better guarantee the rationality and validity of the generated result.

Recently, it has been shown that such a non-differentiable task metric problem can be addressed by employing the techniques from reinforcement learning [31], [33], [36]. In our framework, we design a capability-aware policy gradient training algorithm. Here, we can view the combination of the above CANTM and CANJDG as an *agent*, which interacts with the *environment*, that is the training instances. Given the input job duty $X$, the *policy* $p_{\boldsymbol{\theta}}(\widehat{y_{j,k}}|X, \widehat{y_{<j}}, \widehat{y_{j,<k}})$ is defined by the parameters $\boldsymbol{\theta}$ of the agent, that decide each *action*, i.e., the prediction of next $k$th word $\widehat{y_k}$ in $j$th sentence of job description based on the current *state* (e.g., the hidden states in CANTM and CANJDG, capability-aware attention weights $\alpha^t_{l,j}$ and $\alpha^c_{l,j,k}$, capability-aware generation probability $p_{gen}$, etc). The agent then observes a *reward* when it generates the end-of-sequence (EOS) word in the job description. The training objective is to obtain a policy that minimize the negative expected reward

$$\mathcal{L}^{RL} = -\mathbb{E}_{\widehat{Y}\sim p_{\boldsymbol{\theta}}(\widehat{Y}|X)}[r(\widehat{Y})], \quad (19)$$

where $\widehat{Y} = \{\widehat{y_{1,1}}, \widehat{y_{1,2}}, \ldots, \widehat{y_{N,M^c_N}}\}$ denotes the action sequences and $r(\cdot)$ denotes the reward function. According to the REINFORCE [28], we have

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}^{RL} = -\mathbb{E}_{\widehat{Y}\sim p_{\boldsymbol{\theta}}(\widehat{Y}|X)}[r(\widehat{Y})\nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\theta}}(\widehat{Y}|X)], \quad (20)$$

and it can be approximated using a single Monte-Carlo sample $\widehat{Y}$ derived from from the policy $p_{\boldsymbol{\theta}}$ as follows:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L}^{RL} &\approx -r(\widehat{Y})\nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\theta}}(\widehat{Y}|X) \\ &= -r(\widehat{Y})\nabla_{\boldsymbol{\theta}}\sum_{j=1}^{N}\sum_{k=1}^{M^c_j}\log p_{\boldsymbol{\theta}}(\widehat{y_{j,k}}|X, \widehat{y_{<j}}, \widehat{y_{j,<k}}) \\ &= -r(\widehat{Y})\nabla_{\boldsymbol{\theta}}\sum_{j=1}^{N}(\log p_{\boldsymbol{\theta}}(\widehat{t_j}|X, \widehat{t_{<j}}) \\ &\quad + \sum_{k=1}^{M^c_j}\log p_{\boldsymbol{\theta}}(\widehat{y_{j,k}}|X, \widehat{t_j}, \widehat{y_{<j}}, \widehat{y_{j,<k}})), \end{aligned} \quad (21)$$

where $\{\widehat{t_1}, \ldots, \widehat{t_N}\}$ is a Monte-Carlo sample of the predicted capability label sequence, which derived from $p_{\boldsymbol{\theta}}$. $p_{\boldsymbol{\theta}}(\widehat{t_j}|X, \widehat{t_{<j}})$ and $p_{\boldsymbol{\theta}}(\widehat{y_{j,k}}|X, \widehat{t_j}, \widehat{y_{<j}}, \widehat{y_{j,<k}})$ are calculated by Equations (12) and (15), respectively.

**Algorithm 1.** The Entire Training Process of Cajon

**Input:** The set of job posting $\mathcal{C} = \{C_i = (X_i, Y_i)\}_{i=1}^{|\mathcal{C}|}$.
**Output:** The trained model $\mathcal{M}$ that can generate fluent and rational job description $Y_{new}$ when a new job duty $X_{new}$ is given.
1: Extract the capability words of each job description $Y_i$;
2: Pretrain the neural topic model CANMT with $\{(X^{bow}_i, S^{bow}_i)\}$ by Equation (6);
3: Train the CANMT and CANJDG together with $\{(X_i, X^{bow}_i, Y_i, S_i, S^{bow}_i)\}$ by Equation (25) to obtain $\mathcal{M}$.

In addition, the policy gradient obtained from Equation (21) often suffer from high variance problem. Here, we follow the idea of self-critical policy gradient training algorithm [33] to add a baseline reward, which is calculated by the greedy selection of the generated word at each time step. Here, we use $\widehat{Y^g} = \{\widehat{y^g_{1,1}}, \widehat{y^g_{1,2}}, \ldots, \widehat{y^g_{N,M^c_N}}\}$ denotes the generated job description by greedy search. Following this mechanism, we can update Equation (21) as follows:

TABLE 2
The Statistics of the Dataset

| Datasets | Statistics | Values |
|---|---|---|
| T | # of (job duty, job description) pair | 3,475 |
| | Average words per job duty | 50.11 |
| | Average words per job description | 100.10 |
| | Average capability words per job description | 25.72 |
| | Average sentence per job description | 6.95 |
| P | # of (job duty, job description) pair | 2,351 |
| | Average words per job duty | 74.98 |
| | Average words per job description | 75.80 |
| | Average capability words per job description | 15.58 |
| | Average sentence per job description | 4.98 |

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}^{RL} \approx (r(\widehat{Y^g}) - r(\widehat{Y}))\nabla_{\boldsymbol{\theta}} \sum_{j=1}^{N}(\log p_{\boldsymbol{\theta}}(\widehat{t}_j|X, \widehat{t_{<j}}) + \sum_{k=1}^{M_j^c} \log p_{\boldsymbol{\theta}}(\widehat{y_{j,k}}|X, \widehat{t}_j, \widehat{y_{<j}}, \widehat{y_{j,<k}})). \quad (22)$$

As we mentioned before, we hope to directly optimize the accuracy of the capabilities in the generated job description. Therefore, we choose the F1-value of the generated capability words into the reward function, that is

$$r_{capability}(\widehat{Y}) = \frac{2Card(\widehat{S} \cap S)}{Card(\widehat{S}) + Card(S)}, \quad (23)$$

where $S$ is the set of capability words in the ground-truth job description, $\widehat{S}$ is the set of capability words in $\widehat{Y}$, and $Card(\cdot)$ denotes the size of one set. Moreover, we also take the Rouge-1 score [29] into the reward function, which is used to measure the longest common subsequence based statistics between the ground-truth and generated job description. So that, we can directly optimize the sentence-level structure similarity with the ground-truth, which helps to improve the language fluency. Then, we can set the reward function as

$$r(\widehat{Y}) = \lambda_3 r_{capability}(\widehat{Y}) + \text{Rouge-1}(\widehat{Y}, Y), \quad (24)$$

where $Y$ denotes the ground-truth job description and $\lambda_3$ is a hyperparameter to balance the effects of two different reward scores.

Finally, we mix the $\mathcal{L}^*$ and $\mathcal{L}^{RL}$ to observe the overall learning object function by:

$$\mathcal{L} = (1-\gamma)\mathcal{L}^* + \gamma\mathcal{L}^{RL}, \quad (25)$$

where $\gamma$ is a dynamic hyperparameter during the training process. We first set it to 0 to only train our model by $\mathcal{L}^*$ for a period of time, and then gradually increase its weight to involve the reinforcement learning loss. The entire training process of Cajon is show in Algorithm 1.

## 4 Experimental Results

In this section, we will introduce our extensive experiments with both quantitative analysis and human evaluation on a real-world recruitment dataset to demonstrate the effectiveness of our Cajon.

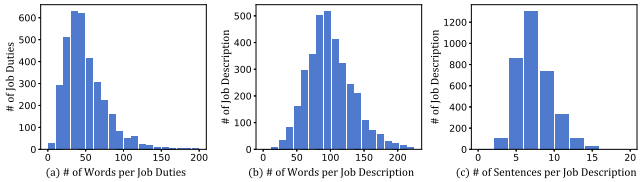



Fig. 5. Distribution of job duty and description inputs of the Technology dataset.

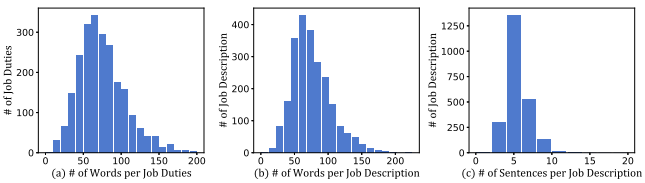
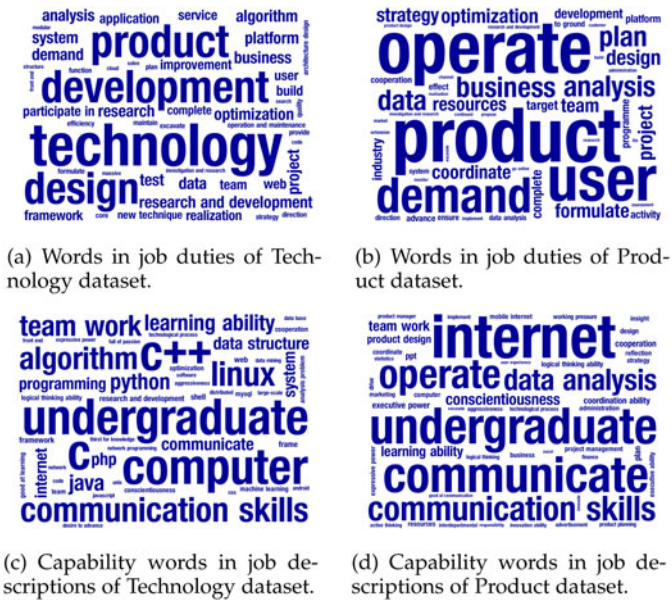



Fig. 6. Distribution of job duty and description inputs of the Product dataset.




(a) Words in job duties of Technology dataset.

(b) Words in job duties of Product dataset.

(c) Capability words in job descriptions of Technology dataset.

(d) Capability words in job descriptions of Product dataset.

Fig. 7. The word cloud representation related to words in job duties and capability words in job descriptions.

*Data Description*. In this paper, we conducted our experiments on two real-world recruitment dataset, which were provided by a high-tech company in China. Specifically, these two datasets are about Technology (T) and Product (P) related job postings. Here, we collected the job duty and corresponding job description documents from 3,475 and 2,351 different job postings, respectively, which have been carefully proofread by the 6 staffing experts in HR department for guaranteeing the fluency and rationality. Some statistics of our dataset are shown in Table 2, Figs. 5, 6, and 7. Because some capability words in job descriptions are in English, such as "Torch" and "Python", we converted all text in English into lowercase. Then, we performed tokenization by using Jieba (Chinese text segmentation). Here, we randomly selected 80% of the dataset as training data, another 10% as test data to validate the performance, and the last 10% for tuning parameters.

### 4.1 Experimental Setup

Moreover, to generate the capability words in job descriptions, we followed the idea in [17] and trained an LSTM-CRF model to extract the possible capability words. Specifically, we labeled skill words of 500 job descriptions in each dataset. After split sentences of the labeled job descriptions, we obtained nearly 6 thousand training instances to learn the LSTM-CRF model. Here, we leverage both character, character bigram, and word representations as to the input, which are pretrained from a large-scale recruitment data. Moreover, to further improve the extraction performance, we employed the Chinese pre-trained language model BERT-WWM representation [57] as the additional input feature. Along this line, we extracted the possible capability words of all the job descriptions. With the help of the staffing experts cleaned the capability words, we finally got a clean capability vocabulary with 4,825 entities. Each job description contains about 25 and 15 capability words on average for different datasets, respectively.

*Training Details.* In our Capability-aware Neural Topic Model (CANTM) model, we first converted the original inputs of job duties and capability words in job descriptions into the bag-of-words vectors. In particular, before cooking job duty bag-of-words inputs, we removed stopwords, and also removed the high frequency and low frequency words to enhance the model performance. We set the topic number $(K^d, K^s)$ as $(30, 50)$ and $(30, 30)$ for the Technology and Product datasets, respectively. Moreover, we added four Batch Normalizing Transforms [59] when we calculated $\mu_d(X_i, S_i)$, $\mu_s(X_i, S_i)$, $\log \sigma_d(X_i, S_i)$ and $\log \sigma_s(X_i, S_i)$ in Equation (5), to avoid Kullback-Leibler divergence vanishing problem during the training process [60].

In our Capability-aware Neural Job Description Generation (CANJDG) model, the embedding layers transfer word $x_k$, $y_{j,k}$ and latent capability topic label $t_k$ as vectors $e^d_k$, $e^c_{j,k}$ and $e^t_k$ of size 128, 128 and 50 respectively. More specifically, we used the Skip-gram model to pre-train the word embedding from all the job duty and job description data, and then utilized the result to initialize the embedding layer weights $W_{e^d}$ and $W_{e^c}$. The sequence encoder were implemented by a bidirectional LSTM with the hidden size of 256 for each LSTM layer. The capability-aware sequence decoder was implemented by two unidirectional LSTM with both hidden sizes of 256 for handling the latent capability topics and words in job description. Moreover, the sizes of hidden states in the capability-aware attention mechanisms and capability-aware copy mechanism were set to 256.

In the training process, we first initialized all the parameters with followed Xavier initialization strategy [61]. Then, we pretrained the CANTM model for 200 epochs separately. And we jointly trained Cajon without $\mathcal{L}^{RL}$ according to Equation (17) with empirically settings $\lambda_1 = 1$ and $\lambda_2 = 1$. Finally, we set $\lambda_3 = 1$ and gradually increased the dynamic hyperparameter $\gamma$ to train our Cajon by Equation (25). Moreover, we applied Adam optimization algorithm [62] with an initial learning rate as 0.001 to train our model. We also set the gradient clipping $= 1.0$ to stabilize the training process. During the generation process in test, we used beam search algorithm with beam size $= 4$.

*Evaluation Metrics.* To evaluate the performance of job description generation, we adopted both automatic and human evaluation.

For automatic evaluation, we chose the standard ROUGE metric [29], including ROUGE-1, ROUGE-2 and ROUGE-L which measure the unigram-overlap, bigram-overlap and Longest Common Subsequence (LCS) based statistics between the ground-truth and generated job description. We also used BLEU metric for evaluation [30], which measures the co-occurrences of n-grams between ground-truth and generated job description. Finally, we employed the Precision, Recall and F1-value of the capability words in generated job descriptions to automatically validate the rationality and validity of the generated result.

For human evaluation, we used the metrics listed below to evaluate the quality of a generated job description:

- *Fluency*. Does the generated job description read smoothly and fluently?
- *Validity*. Does the generated job description contain appropriate capability requirement according to the job duty, so that it can validly recruit suitable talents?

In our experiment, we invited three human resources experts in Baidu to conduct the human evaluation. They required to rate each generated job description with a score from 1 to 5, which corresponded from "very terrible" to "very satisfying" with the generated results on *Fluency* and *Rationality* metrics. Please note that, to fairly evaluate the performances of different models, all generated results are first scrambled and then provided to experts for evaluation, and they do not know each generated job description belong to which model.

*Benchmark Methods.* In order to evaluate the effectiveness of our approach, we implemented several state-of-the-art text generation methods which fit our problem setting:

- *Seq2Seq-Attn* is a classic text-to-text generation model, which was proposed in [58] to achieve the neural machine translation. Here we applied the *concat*-based function to calculate attention score, which is similar to the method proposed in our model.
- *DelNet* is a hierarchical generation model which leverages a two-pass decoder to generate text sequence [26].
- *PGNet* is a variant of Seq2Seq-Attn model, which implements the pointer network and coverage mechanism to handle abstractive text summarization [20].
- *TAKG* is a Seq2Seq based neural keyphrase generation model, which combines the neural topic model to explore the word co-occurrence information [52].
- *UniLM* is a transformer network based language generation model, which was proposed in [27] to achieve the sequence-to-sequence prediction task.

We also chose a state-of-the-art automatically write job postings method:

- *SAMA* is a state-of-the-art job posting generation model, which was proposed in [19]. In order to fairly compare with our model, we removed the input of its side information, such as company size and skill graph.

Moreover, we also compared four variants of our Cajon framework to examine the relative influences of the latent capability topics on different competition:

- *Cajon (w/o RL)* is the variant of Cajon that training without CAPGTA, i.e., directly optimize all the parameters by Equation (17).

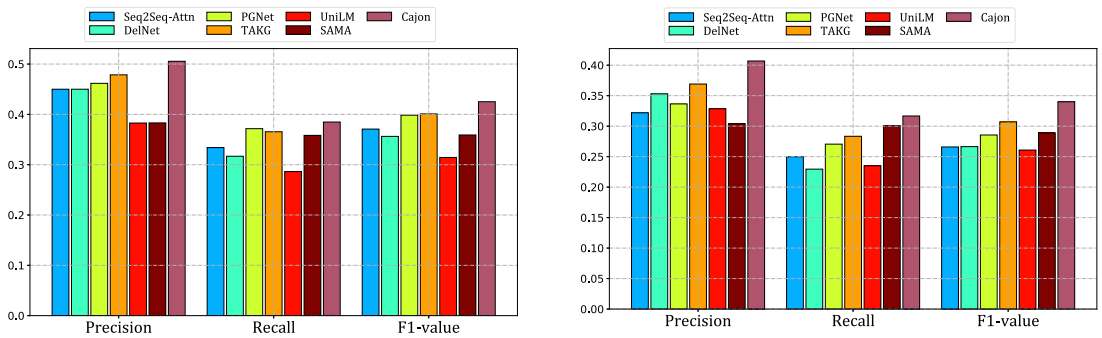
TABLE 3
The Performance of Cajon and Baselines

| Datasets | Methods | ROUGE | | | BLEU | | Human metrics | |
|---|---|---|---|---|---|---|---|---|
| | | ROUGE-1 | ROUGE-2 | ROUGE-L | BLEU-1(%) | BLEU-4(%) | Fluency | Validity |
| T | Seq2Seq-Attn [59] | 0.6389 | 0.4151 | 0.5996 | 37.32 | 16.22 | 3.66 | 3.58 |
| | DelNet [26] | 0.6202 | 0.4015 | 0.5876 | 36.04 | 15.27 | 3.56 | 3.47 |
| | PGNet [20] | 0.6413 | 0.4246 | 0.6061 | 38.14 | 16.75 | 3.70 | 3.61 |
| | TAKG [52] | 0.6460 | 0.4205 | 0.6040 | 39.79 | 16.72 | 3.73 | 3.55 |
| | UniLM [27] | 0.6167 | 0.3965 | 0.5863 | 34.64 | 12.29 | 3.52 | 3.14 |
| | SAMA [19] | 0.6004 | 0.3277 | 0.5455 | 39.30 | 15.07 | 3.47 | 3.52 |
| | Cajon | **0.6531** | **0.4296** | **0.6077** | **41.81** | **19.41** | **3.73** | **3.81** |
| P | Seq2Seq-Attn [59] | 0.7236 | 0.5085 | 0.7138 | 32.02 | 7.28 | 3.63 | 3.34 |
| | DelNet [26] | 0.7292 | 0.5287 | 0.7212 | 30.76 | 6.41 | 3.59 | 3.37 |
| | PGNet [20] | 0.7421 | 0.5410 | 0.7331 | 32.72 | 8.28 | 3.67 | 3.41 |
| | TAKG [52] | 0.7507 | 0.5407 | 0.7372 | 33.89 | 8.27 | 3.72 | 3.50 |
| | UniLM [27] | 0.7542 | 0.5439 | 0.7452 | 31.43 | 6.42 | 3.46 | 3.28 |
| | SAMA [19] | 0.7160 | 0.4788 | 0.7074 | 34.76 | 9.00 | 3.55 | 3.49 |
| | Cajon | **0.7622** | **0.5757** | **0.7513** | **36.46** | **10.84** | **3.78** | **3.74** |

- *Cajon (w/o RL, w LDA)* is the variant of Cajon (w/o RL) that leverages the pretrained LDA on job description data instead of the CANTM.
- *Cajon (w/o RL, $\mathcal{L}_{CANTM}$)* is the variant of Cajon (w/o RL) that does not continue to train the CANTM after loading the pretrained model.
- *Cajon (w/o RL, $\mathcal{L}_{CTL}$)* is the variant of Cajon (w/o RL) that removes latent capability topic label related components in sequence decoder, i.e., it only uses $\theta^s$ to involve the latent capability topic information in sequence decoder.
- *Cajon (w/o RL, topic-attn)* is the variant of Cajon (w/o RL) that removes latent capability topic information, i.e., $\theta^s$, in capability-aware attention mechanism.
- *Cajon (w/o RL, topic-copy)* is the variant of Cajon (w/o RL) that removes capability-aware copy mechanism.

## 4.2 Experimental Results

*Overall Performance.* The comparison results on both automatic and human evaluation for our Cajon framework and baselines are shown in Table 3. According to the results, we can easily see that Cajon outperforms all the baselines with a significant margin. In particular, in the Technology dataset, compared with the best prior arts, we find that our model achieves about 1.84% and 9.62% improvement on the automatic metrics ROUGE-1, BLEU-1 respectively, and 0.81% and 5.54% improvement on human evaluation metrics *Fluency* and *Validity*, respectively. And, in the Product dataset, we find our solution boosts about 1.06% and 4.90% improvement on the automatic metrics ROUGE-1, BLEU-1 respectively, and 3.00% and 7.16% improvement on human evaluation metrics *Fluency* and *Validity*, respectively. This result clearly demonstrates the effectiveness of our model on generating fluent and rational job descriptions. We also found some differences in the order of effects of the baseline methods between ROUGE and BLEU, which due to the different focuses in the calculation of these metrics. And our Cajon can achieve the best results in all automated metrics. In addition, we can note that our model has a greater improvement in the *Validity* on both datasets. This is because our CANTM can effectively learn the capability requirements in the job description and further guide the job descriptions generation. Meanwhile, SAMA can achieve relatively good performance on *Validity*, especially on the Product dataset, because it optimizes skill prediction in the text generation processing. Moreover, we also found that our Cajon performs better than TAKG, which employs a basic neural topic model on a particular text generation task, namely keywords generation. It implies that different from the neural topic model in TAKG, our CANTM can effectively jointly model the topic spaces of

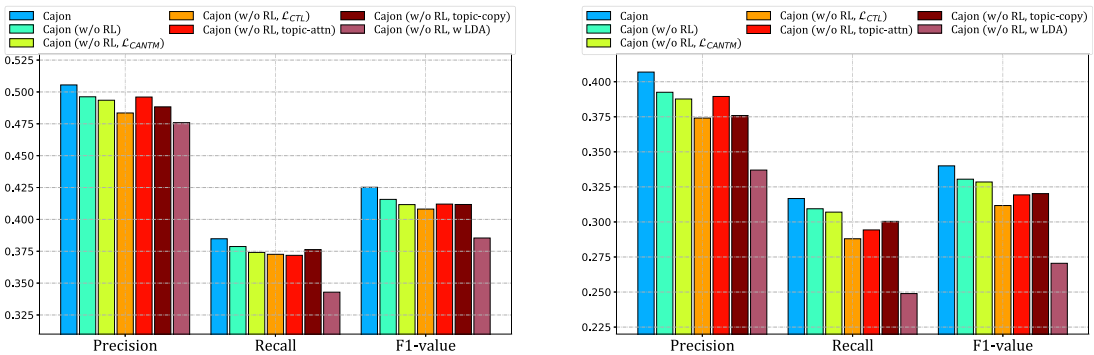



(a) Technology dataset (b) Product dataset

Fig. 8. The Precision, Recall and F1-value of the capability words of Cajon and baselines.

TABLE 4
The Performance of Cajon and its Variants

| Datasets | Methods | ROUGE | | | BLEU | | Human metrics | |
|---|---|---|---|---|---|---|---|---|
| | | ROUGE-1 | ROUGE-2 | ROUGE-L | BLEU-1(%) | BLEU-4(%) | Fluency | Validity |
| T | Cajon | **0.6531** | **0.4296** | **0.6077** | **41.81** | 19.41 | **3.73** | **3.81** |
| | Cajon (w/o RL) | 0.6430 | 0.4177 | 0.5982 | 41.22 | **19.45** | 3.68 | 3.77 |
| | Cajon (w/o RL, w LDA) | 0.6393 | 0.4067 | 0.5939 | 37.93 | 16.00 | 3.58 | 3.63 |
| | Cajon (w/o RL, $\mathcal{L}_{CANTM}$) | 0.6348 | 0.4186 | 0.5954 | 41.15 | 19.19 | 3.65 | 3.75 |
| | Cajon (w/o RL, $\mathcal{L}_{CTL}$) | 0.6327 | 0.4003 | 0.5876 | 41.20 | 18.71 | 3.59 | 3.64 |
| | Cajon (w/o RL, topic-attn) | 0.6365 | 0.3977 | 0.5885 | 41.24 | 17.94 | 3.63 | 3.69 |
| | Cajon (w/o RL, topic-copy) | 0.6411 | 0.4127 | 0.5972 | 41.46 | 19.14 | 3.68 | 3.67 |
| P | Cajon | **0.7622** | **0.5757** | **0.7513** | 36.45 | **10.84** | **3.78** | **3.74** |
| | Cajon (w/o RL) | 0.7598 | 0.5679 | 0.7438 | **37.66** | 10.74 | 3.75 | 3.68 |
| | Cajon (w/o RL, w LDA) | 0.7366 | 0.5291 | 0.7212 | 32.43 | 6.90 | 3.54 | 3.36 |
| | Cajon (w/o RL, $\mathcal{L}_{CANTM}$) | 0.7454 | 0.5421 | 0.7367 | 35.54 | 10.13 | 3.71 | 3.64 |
| | Cajon (w/o RL, $\mathcal{L}_{CTL}$) | 0.7381 | 0.5357 | 0.7279 | 34.02 | 8.93 | 3.64 | 3.47 |
| | Cajon (w/o RL, topic-attn) | 0.7434 | 0.5288 | 0.7312 | 34.77 | 9.45 | 3.66 | 3.51 |
| | Cajon (w/o RL, topic-copy) | 0.7406 | 0.5331 | 0.7330 | 35.90 | 9.55 | 3.70 | 3.54 |




(a) Technology dataset (b) Product dataset

Fig. 9. The Precision, Recall and F1-value of the capability words of Cajon and its variants.

job duties and job description, learn the mapping relationship with distilling the various capability information, and further guide the job description generation in CANJRG.

Then, we calculated the precision, recall and F1-value of the capability words in generated job descriptions, which are shown in Fig. 8. Here we find that our model outperforms the best results of all the baselines about 9.49%, 3.55% and 6.73% in the Technology data, and 20.92%, 5.29% and 17.69% in the Product dataset. It clearly confirms the generated results of our framework could more accurately cover relevant and representative capability requirements for job postings.

Moreover, we evaluated the performance of our Cajon on a public dataset that is proposed in [19]. In order to fit with our problem definition and fairly compare the effects of Cajon and several competitive baselines, we have removed information other than job duty and job description in the dataset, such as skill graph. Meanwhile, because the annotated skill words in this dataset contain some noise (i.e., non-skill words, such as stop words), to achieve better performance of Cajon, we used the words in a filtered skill vocabulary as the input of the capability word list $S_i$ during training. Please noted that we still use the original skill vocabulary to evaluate the performance of capability words prediction. The comparison results are shown in Fig. 10, and we can find that our Cajon achieves the best performance. In addition, we found that some baselines such as SAMA achieved better performance on BLEU in [19] but relatively poor on ROGUE metrics. This may be because we only used the job duty to train all the baselines and used the beam search algorithm during the generation.

*Ablation Analysis.* The main novelty of our model is that it incorporates a CANTM to explore the latent capability topics and apply it into different parts of the neural generation process. Therefore, we compared our model with its four variants as we mentioned before. In addition, Seq2Seq-Attn can actually be regarded as a variant of our model which completely removes CANTM, i.e., removes the effects of $\theta^s$ in Cajon (w/o

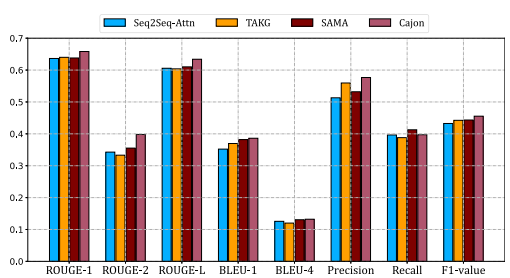



Fig. 10. The performance of the public job posting dataset.

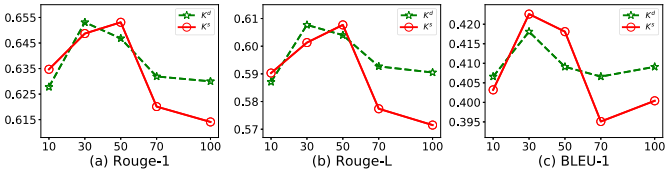



Fig. 11. The impact of topic numbers $K^d$ and $K^s$ in the Technology dataset.

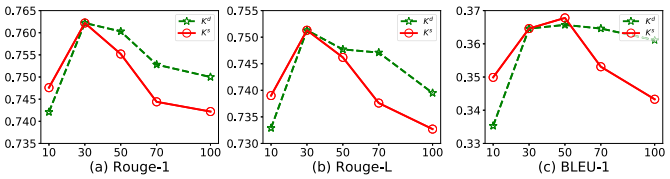



Fig. 12. The impact of topic numbers $K^d$ and $K^s$ in the Product dataset.

$\mathcal{L}_{CTL}$). The results are summarized in Table 4 and Fig. 9. Clearly, we find that all the components are useful to improve the performance. Specifically, we find that when only considering latent capability topic information $\theta^s$, the performances have dropped rapidly, which proves the significance of incorporating the decoder to forecast the latent capability topic label. In addition, we find that the capability-aware attention mechanism can boost about 2.61% and 1.38% on ROUGE-1 and BLEU-1 respectively in the Technology dataset, and boost 2.53% and 4.83% in the Product dataset. We also observe that the capability-aware copy mechanism can boost about 1.87% and 0.84% on ROUGE-1 and BLEU-1 respectively in the Technology dataset, and boost 2.92% and 1.54% in the Product dataset. Those results clearly validate the effectiveness of both mechanisms. Moreover, from Fig. 9, we find that training our model with reinforcement learning has effectively improved the precision, recall and F1-value of the capability words in generated job descriptions, which clearly validates the effectiveness of our capability-aware policy gradient training algorithm (CAPGTA).

*Impact of Latent Capability Topic Number.* To evaluate the parameter sensitivity, we trained our Cajon by varying the two parameters $K^d$ and $K^s$ from 10 to 100 respectively with the other one fixed, i.e., $K^d = 30$ and $K^s = 50$ in the Technology dataset, $K^d = 30$ and $K^s = 30$ in the Product dataset. Figs. 11 and 12 show the results on ROUGE-1, ROUGE-L and BLEU-1 in different datasets. Clearly, we can observe that it can achieve the best performance in term of ROUGE-1 and BLEU-1 when $K^d = 30$ and $K^s = 50$ in the Technology dataset, and $K^d = 30$ and $K^s = 30$ in the Product dataset, which is the reason we chose them in our experiment setting.

### 4.3 Case Study and Discussion

To further illustrate the effectiveness and interpretability of our framework, we present an example of the generated job description by proposed model in Fig. 13. Obviously, the given job duty is to hire a data mining algorithm engineer. We find that the generated results are fluent and includes the capability requirement about education, work experience, data mining algorithms, basic programming language and team collaboration, most of which are mentioned in the

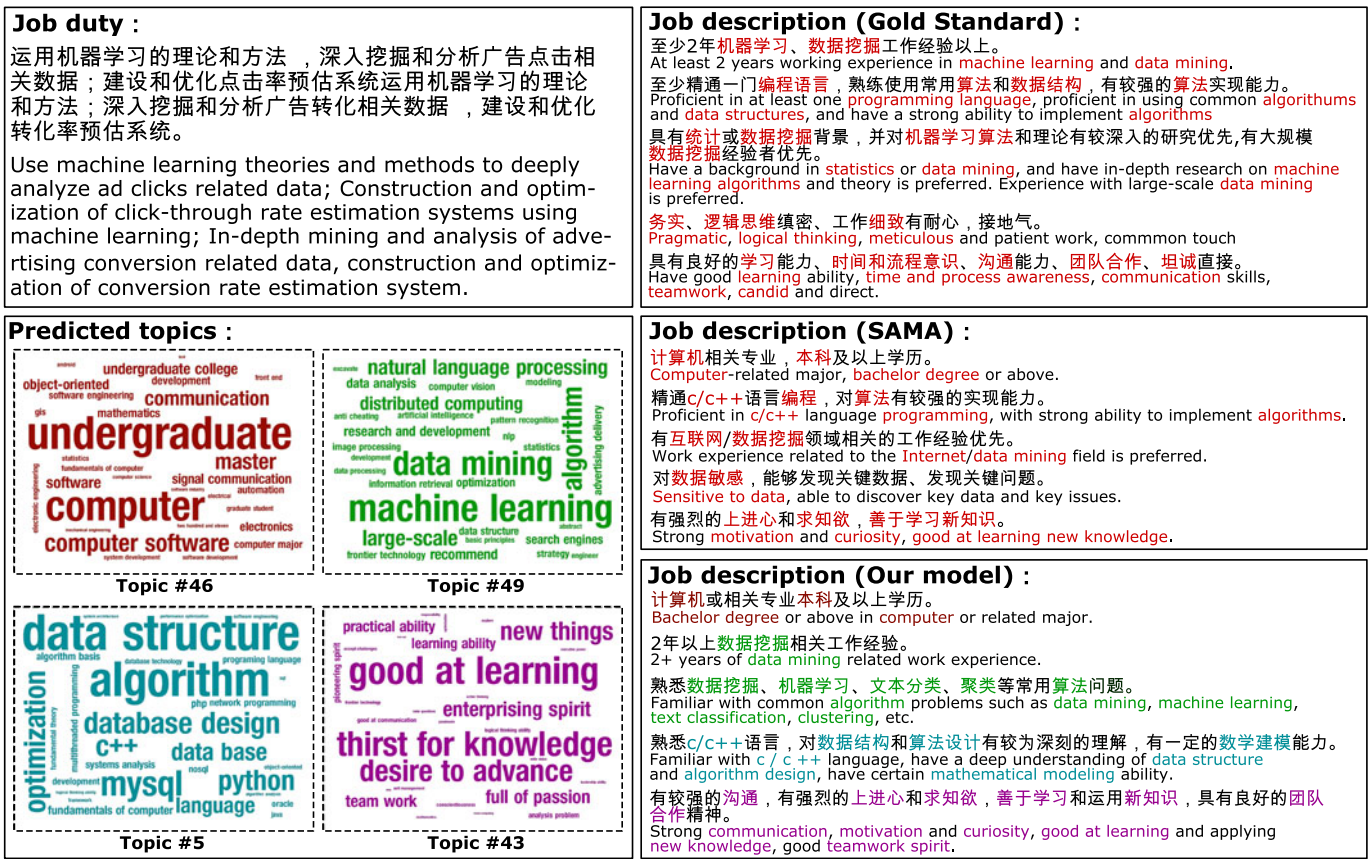



Fig. 13. The case study of job description generation. All the words have been translated into English to better understanding, and the capability words in the ground truth and generated results are highlighted.

ground truth. This proves that our model can effectively generate fluent and rational job description. Moreover, we also show the word cloud representations of the corresponding predicted latent capability topics when generating each job description sentence. Specifically, topic #46 is the education and majors related topic, which is used to generate the first job description sentence. While the topic #49 is about data mining and machine learning, which is used to guide the generation of the second and third job description sentences. In addition, the fourth job description sentence is generated by topic #5, which is about the basic programming language, data structure and algorithms. Finally, the last sentence is generated by topic #43, which is composed of comprehensive qualities such as communication and teamwork ability. It not only demonstrates that our CANTM can effectively learn the meaningful capability topics, but also shows our latent capability topics guide the job description generation, which effectively demonstrates the interpretability of our framework.

## 5 Conclusion

In this paper, we proposed an end-to-end capability-aware neural job description generation framework, namely Cajon, to facilitate the writing of job description. Specifically, we first designed a neural topic model to explore the various capability information from the real-world recruitment data. Then, we introduced an encoder-decoder recurrent neural network to enable job description generation. Then, the capability-aware attention mechanism and copy mechanism were proposed to guide the generation process to ensure the generated job descriptions comprehensively cover relevant and representative capability requirement for job postings. Moreover, we propose a capability-aware policy gradient training algorithm to further enhance the rationality of the generated job description. Finally, extensive experiments conducted on real-world recruitment dataset clearly validated the effectiveness of our Cajon framework and its variants compared with state-of-the-art baselines, as well as its interpretability for learning meaningful capability topics.

## References


[1] Allegis Group, "Let's talk: Focused conversation topics to supercharge recruiting success," 2017. [Online]. Available: https://sc-cms-cdn.azureedge.net/-/media/files/allegisgroup/insights/tas/tas_whitepaper_lets_talk_03_2017_-pdf.pdf

[2] I. E. Solutions, "The-challenges-of-writing-a-job-description," 2013. [Online]. Available: https://www.innovativeemployeesolutions.com/blog/the-challenges-of-writing-a-job-description/

[3] C. Zhu, H. Zhu, H. Xiong, P. Ding, and F. Xie, "Recruitment market trend analysis with sequential latent variable models," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 383–392.

[4] R. P. Schlee and K. R. Harich, "Knowledge and skill requirements for marketing jobs in the 21st century," *J. Marketing Educ.*, vol. 32, no. 3, pp. 341–352, 2010.

[5] C. Qin *et al.*, "Enhancing person-job fit for talent recruitment: An ability-aware neural network approach," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 25–34.

[6] C. Zhu *et al.*, "Person-job fit: Adapting the right talent for the right job with joint representation learning," *ACM Trans. Manage. Inf. Syst.*, vol. 9, no. 3, 2018, Art. no. 12.

[7] X. Wu, T. Xu, H. Zhu, L. Zhang, E. Chen, and H. Xiong, "Trend-aware tensor factorization for job skill demand analysis," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3891–3897.

[8] T. Xu, H. Zhu, C. Zhu, P. Li, and H. Xiong, "Measuring the popularity of job skills in recruitment market: A multi-criteria approach," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, Art. no. 314.

[9] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[10] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[11] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling coverage for neural machine translation," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 76–85.

[12] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 1631–1640.

[13] R. Le, W. Hu, Y. Song, T. Zhang, D. Zhao, and R. Yan, "Towards effective and interpretable person-job fitting," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 1883–1892.

[14] H. Li, Y. Ge, H. Zhu, H. Xiong, and H. Zhao, "Prospecting the career development of talents: A survival analysis perspective," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 917–925.

[15] Q. Meng, H. Zhu, K. Xiao, L. Zhang, and H. Xiong, "A hierarchical career-path-aware neural network for job mobility prediction," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 14–24.

[16] D. Shen, H. Zhu, C. Zhu, T. Xu, C. Ma, and H. Xiong, "A joint learning approach to intelligent job interview assessment," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 3542–3548.

[17] C. Qin *et al.*, "DuerQuiz: A personalized question recommender system for intelligent job interview," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 2165–2173.

[18] M. R. Wade and M. Parent, "Relationships between job skills and performance: A study of webmasters," *J. Manage. Inf. Syst.*, vol. 18, no. 3, pp. 71–96, 2002.

[19] L. Liu, W. Zhang, Z. Chi, W. Shi, and Y. Huang, "Hiring now: A skill-aware multi-attention model for job posting generation," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 3096–3104.

[20] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 1073–1083.

[21] H.-Y. Shum, X.-D. He, and D. Li, "From Eliza to XiaoIce: Challenges and opportunities with social chatbots," *Front. Inf. Technol. Electron. Eng.*, vol. 19, no. 1, pp. 10–26, 2018.

[22] C. Liu, S. He, K. Liu, and J. Zhao, "Vocabulary pyramid network: Multi-pass encoding and decoding with multi-level vocabularies for response generation," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 3774–3783.

[23] E. Reiter and R. Dale, *Building Natural Language Generation Systems*, Cambridge, U.K.: Cambridge Univ. Press, 2000.

[24] L. Zhou and E. Hovy, "Template-filtered headline summarization," in *Proc. ACL Workshop Text Summarization Branches Out*, 2004, pp. 56–60.

[25] M. Elhadad and J. Robin, "An overview of surge: A reusable comprehensive syntactic realization component," in *Proc. Demonstrations Posters Int. Workshop Natural Language Generation*, 1996, pp. 1–4.

[26] Y. Xia *et al.*, "Deliberation networks: Sequence generation beyond one-pass decoding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1784–1794.

[27] L. Dong *et al.*, "Unified language model pre-training for natural language understanding and generation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 13 042–13 054.

[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 229–256, 1992.

[29] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. Workshop Text Summarization Branches Out Post-Conf. Workshop ACL*, 2004, pp. 74–81.

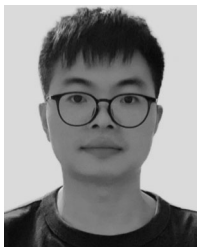
[30] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 311–318.

[31] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," in *Proc. 4th Int. Conf. Learn. Representations*, 2016, pp. 1–16.

[32] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao, "Deep reinforcement learning for dialogue generation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2016, pp. 1192–1202.

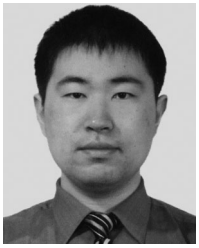
[33] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1179–1195.
[34] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," in *Proc. 5th Int. Conf. Learn. Representations*, 2017, pp. 1–12.
[35] M. Hu, Y. Peng, Z. Huang, X. Qiu, F. Wei, and M. Zhou, "Reinforced mnemonic reader for machine reading comprehension," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 4099–4106.
[36] D. Bahdanau *et al.*, "An actor-critic algorithm for sequence prediction," in *Proc. 5th Int. Conf. Learn. Representations*, 2017, pp. 1–17.
[37] Y.-C. Chen and M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 675–686.
[38] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, no. Jan., pp. 993–1022, 2003.
[39] D. Mimno, H. M. Wallach, J. Naradowsky, D. A. Smith, and A. McCallum, "Polylingual topic models," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2009, pp. 880–889.
[40] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2009, pp. 248–256.
[41] W. X. Zhao *et al.*, "Comparing Twitter and traditional media using topic models," in *Proc. Eur. Conf. Inf. Retrieval*, 2011, pp. 338–349.
[42] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 1445–1456.
[43] T. Kurashima, T. Iwata, T. Hoshide, N. Takaya, and K. Fujimura, "Geo topic model: Joint modeling of user's activity area and interests for location recommendation," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, 2013, pp. 375–384.
[44] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang, "Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 193–202.
[45] Y. Wang, E. Agichtein, and M. Benzi, "TM-LDA: Efficient online modeling of latent topic transitions in social media," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2012, pp. 123–131.
[46] T. Xu, H. Zhu, E. Chen, B. Huai, H. Xiong, and J. Tian, "Learning to annotate via social interaction analytics," *Knowl. Inf. Syst.*, vol. 41, no. 2, pp. 251–276, 2014.
[47] J. D. Mcauliffe and D. M. Blei, "Supervised topic models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2008, pp. 121–128.
[48] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth, "The author-topic model for authors and documents," in *Proc. 20th Conf. Uncertainty Artif. Intell.*, 2004, pp. 487–494.
[49] A. Srivastava and C. Sutton, "Autoencoding variational inference for topic models," in *Proc. 5th Int. Conf. Learn. Representations*, 2017, pp. 1–12.
[50] Y. Miao, E. Grefenstette, and P. Blunsom, "Discovering discrete latent topics with neural variational inference," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2410–2419.
[51] J. Zeng, J. Li, Y. Song, C. Gao, M. R. Lyu, and I. King, "Topic memory networks for short text classification," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2018, pp. 3120–3131.
[52] Y. Wang, J. Li, H. P. Chan, I. King, M. R. Lyu, and S. Shi, "Topic-aware neural keyphrase generation for social media language," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 2516–2526.
[53] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: Understanding rating dimensions with review text," in *Proc. 7th ACM Conf. Recommender Syst.*, 2013, pp. 165–172.
[54] D. Shen, C. Qin, H. Zhu, T. Xu, E. Chen, and H. Xiong, "Joint representation learning with relation-enhanced topic models for intelligent job interview assessment," *ACM Trans. Inf. Syst.*, vol. 40, no. 1, pp. 1–36, 2021.
[55] H. Bahuleyan, L. Mou, O. Vechtomova, and P. Poupart, "Variational attention for sequence-to-sequence models," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 1672–1682.
[56] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.
[57] Y. Cui, W. Che, T. Liu, B. Qin, and Z. Yang, "Pre-training with whole word masking for chinese BERT," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 29, pp. 3504–3514, 2021.
[58] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empir. Methods Natural Lang. Process*, 2015, pp. 1412–1421.
[59] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[60] Q. Zhu, W. Bi, X. Liu, X. Ma, X. Li, and D. Wu, "A batch normalized inference network keeps the KL vanishing away," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 2636–2649.
[61] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[62] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–15.

**Chuan Qin** (Member, IEEE) received the PhD degree in computer science and technology from the University of Science and Technology of China (USTC), Hefei, China, in 2021. He is currently a data scientist with Baidu Inc. He has authored more than 20 journal and conference papers in the fields of natural language processing and recommender system, including the *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Information Systems*, KDD, SIGIR, WWW, NeurIPS, AAAI, IJCAI, ICDM, etc.

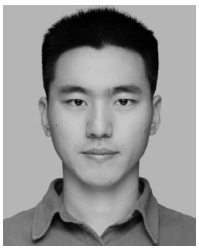


**Kaichun Yao** received the PhD degree from the University of Chinese Academy of Sciences (UCAS), Beijing, China, in 2019. He is currently an algorithm engineer with Baidu Inc. His current research interests include natural language processing, knowledge graph, deep learning, and reinforcement learning. He has published journal and conference papers in fields of artificial intelligence, including the *IEEE Transactions on Cybernetics*, *Neurocomputing*, IJCAI, etc.

**Hengshu Zhu** (Senior Member, IEEE) received the BE and PhD degrees in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2009 and 2014, respectively. He is currently a principal architect & scientist with Baidu Inc. His general area of research is data mining and machine learning, with a focus on developing advanced data analysis techniques for innovative business applications. He has published prolifically in refereed journals and conference proceedings, including the *Proceedings of the IEEE* (PIEEE), *IEEE Transactions on Knowledge and Data Engineering* (TKDE), *IEEE Transactions on Mobile Computing* (TMC), *ACM Transactions on Information Systems* (ACM TOIS), *ACM Transactions on Knowledge Discovery from Data* (TKDD), ACM SIGKDD, ACM SIGIR, WWW, IJCAI, and AAAI. He is a reviewer of many leading academic journals, and has served regularly on the organization and program committees of numerous conferences, including as a program co-chair of the KDD Cup-2019 Regular ML Track, and a founding co-chair of the International Workshop on Organizational Behavior and Talent Analytics (OBTA) and the International Workshop on Talent and Management Computing (TMC), in conjunction with ACM SIGKDD. He was the recipient of the Distinguished Dissertation Award of CAS (2016), Distinguished Dissertation Award of CAAI (2016), Special Prize of President Scholarship for Postgraduate Students of CAS (2014), Best Student Paper Award of KSEM-2011, WAIM-2013, CCDM-2014, and the Best Paper Nomination of ICDM-2014. He is a senior member of the ACM and CCF.



**Tong Xu** (Member, IEEE) received the PhD degree from the University of Science and Technology of China (USTC), Hefei, China, in 2016. He is currently working as an associate professor with the University of Science and Technology of China, Hefei, China. He has authored more than 90 top-tier journal and conference papers in the fields of social network and social media analysis, including the *IEEE Transactions on Image Processing*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Multimedia*, *ACM Transactions on Multimedia Computing, Communications, and Applications*, KDD, SIGIR, WWW, AAAI, IJCAI, etc. He was the recipient of the Best Paper Award of KSEM 2020 and the Best Student Paper Finalist of CICAI 2021.



**Dazhong Shen** received the BS degree in mathematical sciences from the University of Science and Technology of China (USTC), Hefei, China, in 2017. He is currently working toward the PhD degree in the School of Computer Science and Technology, USTC, Hefei, China. His research interests include data mining, topic model, unsupervised learning, and natural language processing.

**Enhong Chen** (Senior Member, IEEE) received the PhD degree from the University of Science and Technology of China (USTC), Hefei, China. He is a professor and vice dean with the School of Computer Science, University of Science and Technology of China (USTC). His research interests include data mining and machine learning, social network analysis, and recommender systems. He has published more than 100 papers in refereed conferences and journals, including the *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Mobile Computing*, KDD, ICDM, NIPS, and CIKM. He was on program committees of numerous conferences including KDD, ICDM, SDM. He received the Best Application Paper Award on KDD-2008, Best Student Paper Award on KDD-2018 (Research), Best Research Paper Award on ICDM2011, and Best of SDM-2015. His research is supported by the National Science Foundation for Distinguished Young Scholars of China.

**Hui Xiong** (Fellow, IEEE) received the PhD degree from the University of Minnesota (UMN), Minneapolis, Minnesota. He is currently a chair professor with the Hong Kong University of Science and Technology, Guangzhou. He is a co-editor-in-chief of the *Encyclopedia of GIS*, an associate editor of the *ACM Transactions on Knowledge Discovery from Data* (TKDD), and *ACM Transactions on Management Information Systems* (TMIS). He has served regularly on the organization and program committees of numerous conferences, including as a program co-chair of the Industrial and Government Track for the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), a program co-chair for the IEEE 2013 International Conference on Data Mining (ICDM), a general co-chair for the IEEE 2015 International Conference on Data Mining (ICDM), and a program co-chair of the Research Track for the 2018 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. He is an ACM distinguished scientist and a fellow of the AAAS.