Constrained Co-embedding Model for User Profiling in Question Answering Communities

Yupeng Luo
School of Data and Computer Science, Sun Yat-sen University, China
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
luoyp21@gmail.com

Shangsong Liang*
School of Data and Computer Science, Sun Yat-sen University, China
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
liangshangsong@gmail.com

Zaiqiao Meng
School of Computing Science, University of Glasgow, Scotland, UK
zaiqiao.meng@gmail.com

ABSTRACT

In this paper, we study the problem of user profiling in question answering communities. We address the problem by proposing a constrained co-embedding model (CCEM). CCEM jointly infers the embeddings of both users and words in question answering communities such that the similarities between users and words can be semantically measured. Our CCEM works with constraints which enforce the inferred embeddings of users and words subject to this criteria: given a question in the community, embeddings of users whose answers receive more votes are closer to the embeddings of the words occurring in these answers, compared to the embeddings of those whose answers receive less votes. Experiments on a Chinese dataset, Zhihu dataset, demonstrate that our proposed co-embedding algorithm outperforms state-of-the-art methods in the task of user profiling.

CCS CONCEPTS

• **Information systems → Question answering**; • **Computing methodologies → Learning latent representations**; • **Mathematics of computing → Variational methods**;

KEYWORDS

Co-embedding; Variational Auto-encoder; User Profiling

1 INTRODUCTION

Question Answering Communities (QAC), e.g., Zhihu [1] and Quora [2], are popular web service platforms, where users post their questions, answer questions posted by other users, and provide feedbacks by, e.g., voting on the questions (thumbs-up or thumbs-down). Understanding the interests of the users is crucial to a number of downstream applications for question answering communities, e.g., automatically inviting users who have expertise on the topics of the questions, and ranking answers to the questions [12, 34, 39]. In this paper, we study the problem of *user profiling*, also called *expert profiling* [2, 21, 22, 25], in question answering communities based on their expertise as expressed in the answers posted by themselves.

The problem of user profiling was first introduced by Balog et al. [2], where language modelling was used to model users' expertise and a set of relevant keywords were identified to represent a user's profile. Similar approaches were applied by Fang and Godavarthy [11] and Rybak et al. [35]. However, these methods suffer from a number of major problems: (1) they regard words as atomic units, resulting in vocabulary mismatch that harms the performance; (2) they represent users and words in disjoint vocabulary spaces, making them impossible to measure the similarity between users and words when identifying keywords to profile a user; (3) rich additional information such as thumbs-up/thumbs-down is not utilized to further enhance the performance in these models, and how to integrate this information into the models is still unknown.

To alleviate the aforementioned problems, we propose a **Constrained Co-embedding Model** for user profiling in question answering communities, abbreviated as CCEM. Instead of directly modelling words as atomic units, CCEM targets at building user profiles by embedding users and words in the same semantic space. Unlike most embedding algorithms that infer one category of entities only, such as word2vec [30] and some contextualized embedding approaches [1, 32] (they infer words' embeddings only) and node2vec [6, 14, 19, 33] (they infer nodes' embeddings only), CCEM aims to jointly learn representations of two categories of entities, i.e., users and words, in the same semantic space via the proposed variational auto-encoder method, such that the similarities between them can be effectively measured. To obtain better representations of users and words for user profiling in question answering communities, CCEM fully utilizes the unique auxiliary information available in the communities. Specifically, it adds constraints to enforce the embeddings of users and words subject to the criteria during the inference of the embeddings of users and words: given a question in the community, embeddings of users whose answers receive more votes should be closer to the embeddings of the words appearing in these answers than the embeddings of those whose answers receive less votes.

Our contribution can be summarized as follows:

(1) We propose a constrained co-embedding model, CCEM, that can jointly infer user and word representations in the same

*Shangsong Liang is the corresponding author of the paper.
[1]https://www.zhihu.com
[2]https://www.quora.com

CIKM '19, November 3–7, 2019, Beijing, China
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6976-3/19/11...$15.00
https://doi.org/10.1145/3357384.3358056

semantic space, such that the semantic similarity between users and words can be effectively measured for users profiling.

(2) We propose an inference method based on variational auto-encoder to infer high-quality embeddings of both users and words. It consists of an inference model that maps the input entities of words and users into two Gaussian distributions, and a generative model, which can effectively reconstruct the input based on the semantic represent of users and words.
(3) Our CCEM adds constraints to enforce the embeddings of users and words subject to specific criteria in question answering communities. To the best of our knowledge, we are the first to introduce constraints into variational auto-encoder framework during the inference of the embeddings.
(4) We conduct experiments on dataset crawled from the Zhihu community, which is one of the most popular question answering communities in China, to evaluate the effectiveness of our proposed model in the task of user profiling.

2 RELATED WORK

In this section, we review the work related to ours, including user profiling, variational auto-encoders and word embeddings.

2.1 User Profiling

User profiling has been gaining attention since the expert finding task of the TREC 2005 enterprise track [7, 23] was launched. Balog et al. [2, 3] proposed a generative language modelling approach to address the user profiling task based on a heterogeneous corpus made up of a large organization's intranet. Fang and Godavarthy targeted to the problem of dynamic use profiling, and proposed a probabilistic model to characterize the dynamics of personal expertise [11], where authors' academic publications were used to learn the personal research interests. An evaluation method was proposed in [9] for user profiling, which allows for weighted non-exact matches between system-produced and ground-truth keywords. More recently, Liang et al. [21, 22, 25] proposed to profile social users in the context of streaming short documents in Twitter. All these user profiling algorithms are designed for platforms such as enterprise search [2] and academic web service [11, 26] but not for community question answering. In community question answering platforms, accurate profiling for users can boost the performance of answer ranking, expert finding and tag recommendation. Unfortunately, to the best of our knowledge, there is no user profiling algorithm that targets at the platforms of community question answering.

2.2 Variational Auto-encoders

Variational Auto-encoders (VAEs) [17] have become one of the most popular and powerful generative models in recent years, with its excellent performance and interpretability. VAEs learn latent representations for observed entities by optimizing a lower bound of the log evidence likelihood and by using the reparameterization tricks [17] and Stochastic Gradient Variational Bayes (SGVB) estimator, VAEs can be trained by two types of neural networks: an inference network that maps visible variable to latent representations and a generative network that reconstructs the latent representations to the corresponding observed variables. Many variants of VAEs have been proposed [5, 10, 18, 27, 28], which have been extensively studied and applied in various tasks such as semi-supervised learning [18], image generation [10], sentence generation [5] and text processing [28]. In this work, in order to obtain high quality embeddings of both users and words, we fully utilizes the unique auxiliary information available in question answering communities, i.e., the number of thumbs-up for each answer, and adopt additional constraints into the optimization of VAEs to enforce the embeddings of users and words subject to a specific criteria. To the best of our knowledge, we are the first attempt to adopt constraints during the learning of VAEs.

2.3 Word Embeddings

Word embedding is a task that uses massive amounts of textual data to map semantic information of words into low-dimensional vectors. Word embedding techniques have become hot research topics since the word2vec model had been proposed [30], which leverages local data, i.e., words' context, to learn syntactic and semantic relationships between words. Some other extended work, such as those proposed in [16, 25, 29], also follow the same basic principle. Besides, other ways of generating embeddings have surfaced, which leverage global information to arrive at vector representations for words, such as those proposed in [20, 31]. More recently, some contextualized embedding approach have been proposed to lean word representations by using words' characteristics and polysemy [1, 32]. To the best of our knowledge, we are the first to jointly model the embeddings of two categories of entities, i.e., users and words, in the community question answering scenarios.

3 NOTATIONS AND TASK FORMULATION

In this section, we introduce the main notations used across the whole paper and formally define the task we study.

3.1 Notations

In this paper, we denote scalars by normal letters (e.g., N represents the number of words) while the calligraph typeface letters are used to represent sets (e.g., $\mathcal{W}$ represents a set of words). Matrices are denoted by uppercase and bold letters (e.g., $\mathbf{W}$ represent the word-to-word co-occurrence matrix). The i-th row of matrix, i.e., a vector, is denoted by the bold letter with a subscript (e.g., $\mathbf{W}_i$ is the i-th row of $\mathbf{W}$). A vector with a subscript represents a scalar element of that vector (e.g., $\mathbf{W}_{ij}$ is the element at the i-th row and j-th column of $\mathbf{W}$). We summarize the main notations in Tab. 1.

Let $\mathcal{U}$ be a set of users and $\mathcal{A}_{\mathcal{U}}$ be a set of answers generated by the users in response to the questions in a QAC. Here, $\mathcal{A}_{\mathcal{U}} = \{\mathcal{A}_{u_1}, \mathcal{A}_{u_2}, \ldots, \mathcal{A}_{u_F}\}$ with $\mathcal{A}_u$ being a set of answers generated by user u in response to the questions she answered. Note that in some QACs, such as Zhihu, all the questions are visible for all the users so that every user can make an answer to them, but for privacy consideration all the questions are set to be anonymous.

3.2 Task Formulation

The user profiling task that we address in this paper is defined as follows: given a set of users $\mathcal{U}$, their answers $\mathcal{A}_{\mathcal{U}}$ and the voting

Table 1: Main notations in our paper.

Symbol	Description
$\mathcal{W}$	set of words
$\mathcal{U}$	set of users
$\mathcal{E}^{\mathcal{W}}$	set of pairs between any two words
$\mathcal{E}^{\mathcal{U}}$	set of pairs between users and words
$N = \lvert\mathcal{W}\rvert$	number of words
$F = \lvert\mathcal{U}\rvert$	number of users
D	dimension of latent variables
$\mathbf{W} \in \mathbb{R}^{N\times N}$	word-to-word co-occurrence matrix
$\mathbf{V} \in \mathbb{R}^{F\times N}$	user-to-word co-occurrence matrix
$\mathbf{Y}$	latent representation matrix for all the *words*
$\mathbf{Z}$	latent representation matrix for all the *users*
$\mathcal{K}_{\mathcal{U}}$	set of top-k relevant keywords for all the *users*

information of these answers $\mathcal{T}_{\mathcal{A}}$, learn semantic representations of both users and words, and then output a ranked list with top-K relevant keywords as the profile for each user. The user profiling algorithm is essentially a function f that satisfies the following:

$$\mathcal{U}, \mathcal{A}_{\mathcal{U}}, \mathcal{T}_{\mathcal{A}_{\mathcal{U}}} \xrightarrow{f} \mathcal{K}_{\mathcal{U}}, \tag{1}$$

where $\mathcal{T}_{\mathcal{A}_{\mathcal{U}}} = \{t_{a_1}, t_{a_2}, \ldots, t_{|\mathcal{T}_{\mathcal{A}_{\mathcal{U}}}|}\}$ is the voting information with t_a being the number of the accumulated votes for the answer a, $\mathcal{K}_{\mathcal{U}} = \{\mathcal{K}_{u_1}, \mathcal{K}_{u_2}, \ldots, \mathcal{K}_{u_F}\}$ are all the users' profiling results with $\mathcal{K}_u = \{w_{u,1}, w_{u,2}, \ldots, w_{u,K}\}$ being the profiling result, i.e., the top-K relevant keywords for profiling the user u.

4 CONSTRAINED CO-EMBEDDING MODEL

In this section, we introduce our constrained co-embeddings model, CCEM, that arms at jointly inferring embeddings of both users and words in the same semantic space such that the similarities between them can be effectively measured for user profiling.

4.1 Preliminaries

The goal of CCEM is to learn the semantic representations for both users and and words, i.e. $\mathbf{Z} \in \mathbb{R}^{F\times D}$ with $\mathbf{Z}_u$ being user u' embedding and $\mathbf{Y} \in \mathbb{R}^{N\times D}$ with Y_w being word w' embedding, respectively, where N, F and D denote the size of the vocabulary and the number of users and the dimension of embeddings, respectively.

Our CCEM model is based on the well-known VAE model [17]. VAE consists of two neural networks: an inference network $q_\phi(z|x)$, also called *Encoder*, which infers the observed variables to the latent variables which approximate a Gaussian prior $p(z)$, and a generative network $p_\theta(x|z)$, also called *Decoder*, which samples the visible variables given the latent variables. The objective of VAE is to optimize the Evidence Lower BOund, i.e., ELBO, of $\log p_\theta(x)$:

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - D_{KL}(q_\phi(z|x)||p_\theta(x)), \tag{2}$$

where $D_{KL}(\cdot||\cdot)$, ϕ and θ are the KL-divergence, the parameters of the encoder $q_\phi(z|x)$ and the decoder $p_\theta(x|z)$, respectively.

However, there are two limitations when applying this model to the embedding problem in QACs: it only embeds one category of entity while in our setting we have users and words; it is nontrivial to integrate voting information into the learning model.

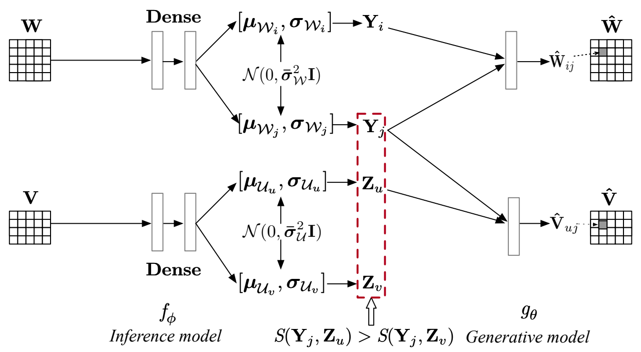

W
Dense
$[\boldsymbol{\mu}_{\mathcal{W}_i}, \boldsymbol{\sigma}_{\mathcal{W}_i}]$
$\mathbf{Y}_i$
$\mathcal{N}(0, \bar{\boldsymbol{\sigma}}^2_{\mathcal{W}}\mathbf{I})$
$[\boldsymbol{\mu}_{\mathcal{W}_j}, \boldsymbol{\sigma}_{\mathcal{W}_j}]$
$\mathbf{Y}_j$
$\hat{\mathbf{W}}$
$\hat{\mathbf{W}}_{ij}$
V
$[\boldsymbol{\mu}_{\mathcal{U}_u}, \boldsymbol{\sigma}_{\mathcal{U}_u}]$
$\mathbf{Z}_u$
$\mathcal{N}(0, \bar{\boldsymbol{\sigma}}^2_{\mathcal{U}}\mathbf{I})$
Dense
$[\boldsymbol{\mu}_{\mathcal{U}_v}, \boldsymbol{\sigma}_{\mathcal{U}_v}]$
$\mathbf{Z}_v$
$\hat{\mathbf{V}}$
$\hat{\mathbf{V}}_{uj}$
f_ϕ
Inference model
$S(\mathbf{Y}_j, \mathbf{Z}_u) > S(\mathbf{Y}_j, \mathbf{Z}_v)$
g_θ
Generative model

Figure 1: The architecture of our CCEM. The inference model f_ϕ and the generative model g_θ act as the probabilistic encoder and the probabilistic decoder, respectively. The model takes the word-to-word co-occurrence matrix W and the user-to-word co-occurrence matrix U as input and maps them to Gaussian distributions with means and variances as latent embeddings for all words and users, respectively. The constraints between Z_u, Z_v, and Y_i which denote as $S(Y_i, Z_u) > S(Y_i, Z_v)$ are adopted to pull users closer to the words they have expertise on in the semantic space.

4.2 Overview of Our Model

Our CCEM consists of three main parts: (I) construct the observed matrices and constraint information; (II) infer users and words embeddings; (III) profile users' expertise; see Algorithm 1 for an overview. In "Part I" of Algorithm 1, we first construct the observed matrices and constraints as the input of CCEM (steps 2-4 in Algorithm 1); In "Part II" of Algorithm 1, CCEM co-embeds users and words into the same space with constraints by integrating VAE and SVM-Rank method, such that the latent representations of users and words can be effectively inferred (steps 6-21); See Fig. 1 for an overview of the architecture of our CCEM. In "Part III", we uses the inferred embeddings produced by CCEM to profile users' expertise with top-K relevant keywords for each user (steps 23-25).

In the following subsections, we first detail the co-embedding model for both users and words (§4.3) and then detail how to incorporate voting constraints into our model (§4.4).

4.3 Co-embedding Users and Words

To obtain embeddings for both users and words, i.e., $\mathbf{Y}$ and $\mathbf{Z}$, we first construct the observed word-to-word and user-to-word co-occurrence matrices, $\mathbf{W}$ and $\mathbf{U}$, in the following way: (1) for building $\mathbf{W}$, we first take the answers of all users as corpus, and then calculating the co-occurrence frequency between two words by setting a sliding window with fixed size; (2) for building $\mathbf{U}$, we build it by counting the co-occurrence information between users and words. Specifically, the co-occurrence information can be split into two categories, which will be discussed in §5.1. We then define an objective to maximize the log-likelihood of the observed matrices, $\mathbf{W}$ and $\mathbf{V}$. By using Jensen's inequality, the log-likelihood of $\mathbf{W}$ and $\mathbf{V}$ can be represented as:

$$\log p(\mathbf{W}, \mathbf{V}) = \log \int_{\mathrm{Y}} \int_{\mathrm{Z}} p(\mathbf{W}, \mathbf{V}, \mathrm{Y}, \mathrm{Z}) \mathrm{d}\mathrm{Y}\mathrm{d}\mathrm{Z}$$

Algorithm 1: Overview of CCEM for user profiling.

```
Input  : A set of users $\mathcal{U}$
         A set of answers $\mathcal{A}_{\mathcal{U}}$ (generated by $\mathcal{U}$)
         Voting information $\mathcal{T}_{\mathcal{A}}$
Output : A set of top-K keywords $\mathcal{K}_{\mathcal{U}}$ for all users
1  /* Part I: Construct the observed matrices and constraints information */
2  Construct matrix W by counting word-to-word co-occurrence information
3  Construct matrix U by counting user-to-word co-occurrence information
4  Construct constraints, $C_w = \{C_{w_1}, C_{w_2}, ..., C_{w_N}\}$, for each word by using voting information $\mathcal{T}_{\mathcal{A}}$
5  /* Part II: Infer users and words embeddings by CCEM */
6  $\theta^{(0)}$,$\phi^{(0)}$ ← Initialize network parameters
7  $w^{(0)}$,$b^{(0)}$ ← Initialize SVM-Rank ( similarity computing function $S(\cdot,\cdot)$ ) parameters as 0
8  repeat
9      $\epsilon_w \leftarrow \mathcal{N}(\mathbf{0}, \bar{\sigma}^2_{\mathcal{W}}\mathbf{I})$
10     $\epsilon_u \leftarrow \mathcal{N}(\mathbf{0}, \bar{\sigma}^2_{\mathcal{U}}\mathbf{I})$
11     for $i = 1, \dots, N$ do
12         for $u, v \in C_{w_i}$ do
13             $\mathrm{Y}^{(t)}, \mathbf{Z}^{(t)} \leftarrow \theta^{(t)}$
14             construct constraints
               $\mathbf{C}^{\mathbf{i}}_{\mathbf{u,v}} = S(\mathrm{Y}_i, \mathbf{Z}_u) - S(\mathrm{Y}_i, \mathbf{Z}_v)$
15     $g_\theta, g_\phi \leftarrow \nabla_{\theta,\phi}\mathcal{L}(\theta, \phi, \mathbf{W}, \mathbf{U}, \epsilon_w, \epsilon_u) - \alpha \sum_{i=1}^{N} \sum_{(u,v)\in c_i} \mathbf{C}^{\mathbf{i}}_{\mathbf{u,v}}$
16     $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta g_\theta$
17     $\phi^{(t+1)} \leftarrow \phi^{(t)} - \eta g_\phi$
18     $\mathrm{Y}^{(t+1)}, \mathbf{Z}^{(t+1)} \leftarrow \theta^{(t+1)}$
19     Construct training sample for SVM-Rank (e.g., $\{x :< \mathrm{Y}_i^{(t+1)}, \mathbf{Z}_u^{(t+1)} > - < \mathrm{Y}_i^{(t+1)}, \mathbf{Z}_v^{(t+1)} >, y : 1\}$)
20     $w^{(t+1)}, b^{(t+1)}$ ← optimize SVM-Rank to obtain parameters of similarity computing function
21 until CCEM converges
22 /* Part III: profile users' expertise */
23 for $u = 1, \dots, F$ do
24     $\mathcal{K}_u$ ← find top-K relevant words using embeddings produced by CCEM
25 Get the set of user profile $\mathcal{K}_{\mathcal{U}} = \{\mathcal{K}_{u_1}, \mathcal{K}_{u_2}, \dots, \mathcal{K}_{u_F}\}$
```

$$= \log \int_{\mathrm{Y}} \int_{\mathbf{Z}} p(\mathbf{W}, \mathbf{V}, \mathrm{Y}, \mathbf{Z}) \frac{q_\phi(\mathrm{Y}, \mathbf{Z} \mid \mathbf{W}, \mathbf{V})}{q_\phi(\mathrm{Y}, \mathbf{Z} \mid \mathbf{W}, \mathbf{V})} \mathrm{dYdZ}$$
$$\geq \mathbb{E}_{q_\phi(\mathrm{Y},\mathbf{Z}|\mathbf{W},\mathbf{V})} \left[\log \frac{p(\mathbf{W}, \mathbf{V}, \mathrm{Y}, \mathbf{Z})}{q_\phi(\mathrm{Y}, \mathbf{Z} \mid \mathbf{W}, \mathbf{V})} \right], \quad (3)$$

where $p(\mathbf{W}, \mathbf{V}, \mathrm{Y}, \mathbf{Z})$ is the joint distribution of latent variables and observations, with Y and Z being all words' embeddings and users' embeddings, respectively, and $q_\phi(\mathrm{Y}, \mathbf{Z} \mid \mathbf{W}, \mathbf{V})$ is the variational posterior over words and users for approximating the true posterior of latent variables $p_\theta(\mathrm{Y}, \mathbf{Z} \mid \mathbf{W}, \mathbf{V})$, with ϕ being the variational parameters to be estimated. For convenient discussion, we abbreviate $q_\phi(\mathrm{Y}, \mathbf{Z} \mid \mathbf{W}, \mathbf{V})$ as q_ϕ in the rest of this paper.

For the joint distribution, we assume the latent variables of the words and users are independent, so that the formulation can be rewritten as:

$$p(\mathbf{W}, \mathbf{V}, \mathrm{Y}, \mathbf{Z}) = p(\mathrm{Y})p(\mathbf{Z}) \prod_{(i,j)\in\mathcal{E}_{\mathcal{W}}} p_{\theta_1}(\mathbf{W}_{ij} \mid \mathrm{Y}_i, \mathrm{Y}_j) \prod_{(u,i)\in\mathcal{E}_{\mathcal{U}}} p_{\theta_2}(\mathbf{V}_{ui} \mid \mathbf{Z}_u, \mathrm{Y}_i), \quad (4)$$

where θ_1 and θ_2 represents the generative parameters, $\mathbf{W}_{ij}$ is the co-occurrence information between word i and word j, with Y_i and Y_j being the corresponding word embeddings, and $\mathbf{V}_{ui}$ is the co-occurrence information between user u and word i, with $\mathbf{Z}_u$ and Y_i being the corresponding user embeddings and word embeddings, respectively.

For the variational posterior q_ϕ, by applying the mean-field assumption, we can rewrite it as follows:

$$q_\phi(\mathrm{Y}, \mathbf{Z} \mid \mathbf{W}, \mathbf{V}) = \prod_{i\in\mathcal{W}} q_{\phi_1}(\mathrm{Y}_i \mid \mathbf{W}) \prod_{u\in\mathcal{U}} q_{\phi_2}(\mathbf{Z}_u \mid \mathbf{V}), \quad (5)$$

where ϕ_1 and ϕ_2 represent the inference parameters of words and users, respectively. Substituting Eq. 4 and Eq. 5 into Eq. 3 with a trade-off parameter β, Eq. 3 can be represented as:

$$\log p(\mathbf{W}, \mathbf{V}) \geq \mathcal{L}(\theta_1, \phi_1; \mathbf{W}) + \beta\mathcal{L}(\theta_2, \phi_2; \mathbf{W}, \mathbf{V}) \triangleq \mathcal{L}(\theta, \phi; \mathbf{W}, \mathbf{V}), \quad (6)$$

where

$$\mathcal{L}(\theta_1, \phi_1; \mathbf{W}) = \mathbb{E}_{q_\phi} \left[\sum_{(i,j)\in\mathcal{E}_{\mathcal{W}}} \log p_{\theta_1}(\mathbf{W}_{ij} \mid \mathrm{Y}_i, \mathrm{Y}_j) \right] - D_{KL}(q_{\phi_1}(\mathrm{Y} \mid \mathbf{W}) \parallel p(\mathrm{Y})), \quad (7)$$

$$\mathcal{L}(\theta_2, \phi_2; \mathbf{W}, \mathbf{V}) = \mathbb{E}_{q_\phi} \left[\sum_{(u,i)\in\mathcal{E}_{\mathcal{U}}} \log p_{\theta_2}(\mathbf{V}_{ui} \mid \mathbf{Z}_u, \mathrm{Y}_i) \right] - D_{KL}(q_{\phi_2}(\mathbf{Z} \mid \mathbf{V}) \parallel p(\mathbf{Z})), \quad (8)$$

where $\mathcal{L}(\theta, \phi; \mathbf{W}, \mathbf{V})$ is the Evidence Lower BOund (ELBO) on the marginal likelihood of the observed variables,which include word-to-word part, i.e. $\mathcal{L}(\theta_1, \phi_1; \mathbf{W})$, and user-to-word part, i.e., $\mathcal{L}(\theta_2, \phi_2; \mathbf{W}, \mathbf{V})$, and $D_{KL}(\cdot \parallel \cdot)$ is the KL-divergence. Further more, $q_{\phi_1}(\mathrm{Y} \mid \mathbf{W})$ and $q_{\phi_2}(\mathbf{Z} \mid \mathbf{V})$ are referred to the inference model, since given the observed data, i.e., W and V, they aim at producing variational posterior distributions over the possible values of the latent embeddings Y and Z. Similarly, $p_{\theta_1}(\mathbf{W}_{ij} \mid \mathrm{Y}_i, \mathrm{Y}_j)$ and $p_{\theta_2}(\mathbf{V}_{ui} \mid \mathbf{Z}_u, \mathrm{Y}_i)$ are referred to the generative model, since given the latent embeddings Y and Z, they produce a distribution over the possible corresponding values of observed users and words. The KL-divergence terms can be interpreted as the regularizer, while the expectation terms are regarded as expected negative reconstruction error loss.

Analytical solutions of expectations w.r.t. the variational posterior are intractable in general cases, but we can reduce the problem of estimating the gradient w.r.t. parameters of the posterior distribution to a simpler problem of estimating the gradient w.r.t.

parameters of a deterministic function, which is called the *reparameterization* trick [17].

Similar to VAE [17], we assume all of the priors of latent variables and the variational posterior distributions to be Gaussian:

$$p(\mathbf{Y}) = \mathcal{N}(\mathbf{0}, \bar{\sigma}^2_{\mathcal{W}}\mathbf{I}), \tag{9}$$

$$p(\mathbf{Z}) = \mathcal{N}(\mathbf{0}, \bar{\sigma}^2_{\mathcal{U}}\mathbf{I}), \tag{10}$$

$$q_{\phi_1}(\mathbf{Y}_i \mid \mathbf{W}) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{W}_i}, \boldsymbol{\sigma}^2_{\mathcal{W}_i}\mathbf{I}), \tag{11}$$

$$q_{\phi_2}(\mathbf{Z}_u \mid \mathbf{V}) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{U}_u}, \boldsymbol{\sigma}^2_{\mathcal{U}_u}\mathbf{I}), \tag{12}$$

where $\bar{\sigma}^2_{\mathcal{W}}$, $\bar{\sigma}^2_{\mathcal{U}}$ are hyperparameters of the priors, $\boldsymbol{\mu}_{\mathcal{W}_i}$ and $\boldsymbol{\mu}_{\mathcal{U}_u}$ $\boldsymbol{\sigma}^2_{\mathcal{W}_i}$ and $\boldsymbol{\sigma}^2_{\mathcal{U}_u}$ are means and variances of word embeddings and user embeddings to be learned, respectively.

Since we assume the priors and the variational posteriors are Gaussian distributions, the KL-divergence terms in Eq. 6 can be obtained in analytical forms. By using the SGVB estimator and the reparameterization trick, we can directly derivate from Monte Carlo estimates of these expectation terms by the following estimators of this model:

$$\begin{aligned}\mathcal{L}(\theta, \phi; \mathbf{W}, \mathbf{V}) &= \frac{1}{N^2 \cdot L}\sum_{l=1}^{L}\Big(\sum_{i,j\in\mathcal{W}} \log p_{\theta_1}(\mathbf{W}_{ij} \mid \mathbf{Y}_i{}^{(l)}, \mathbf{Y}_j{}^{(l)})\Big)\\ &+ \frac{1}{N\cdot F\cdot L}\sum_{l=1}^{L}\Big(\sum_{u\in\mathcal{U}, i\in\mathcal{W}} \log p_{\theta_2}(\mathbf{V}_{ui} \mid \mathbf{Z}_u{}^{(l)}, \mathbf{Y}_i{}^{(l)})\Big)\\ &+ \frac{1}{2N}\sum_{i\in\mathcal{W}}\sum_{d=1}^{D}\Big(1 + \log(\frac{\boldsymbol{\sigma}^2_{\mathcal{W}_i}|_d}{\bar{\boldsymbol{\sigma}}^2_{\mathcal{W}_i}|_d}) - \frac{(\boldsymbol{\mu}_{\mathcal{W}_i}|_d)^2}{\bar{\boldsymbol{\sigma}}^2_{\mathcal{W}_i}|_d} - \boldsymbol{\sigma}^2_{\mathcal{W}_i}|_d \cdot \bar{\boldsymbol{\sigma}}^2_{\mathcal{W}_i}|_d\Big)\\ &+ \frac{1}{2F}\sum_{u\in\mathcal{U}}\sum_{d=1}^{D}\Big(1 + \log(\frac{\boldsymbol{\sigma}^2_{\mathcal{U}_u}|_d}{\bar{\boldsymbol{\sigma}}^2_{\mathcal{U}_u}|_d}) - \frac{(\boldsymbol{\mu}_{\mathcal{U}_u}|_d)^2}{\bar{\boldsymbol{\sigma}}^2_{\mathcal{U}_u}|_d} - \boldsymbol{\sigma}^2_{\mathcal{U}_u}|_d \cdot \bar{\boldsymbol{\sigma}}^2_{\mathcal{U}_u}|_d\Big),\end{aligned}$$

where

$$\begin{aligned}\mathbf{Y}_i{}^{(l)} &= \boldsymbol{\mu}_{\mathcal{W}_i} + \boldsymbol{\sigma}^2_{\mathcal{W}_i} \odot \boldsymbol{\epsilon}^{(l)}, \text{with } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}),\\ \mathbf{Y}_j{}^{(l)} &= \boldsymbol{\mu}_{\mathcal{W}_j} + \boldsymbol{\sigma}^2_{\mathcal{W}_j} \odot \boldsymbol{\epsilon}^{(l)}, \text{with } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}),\\ \mathbf{Z}_u{}^{(l)} &= \boldsymbol{\mu}_{\mathcal{U}_u} + \boldsymbol{\sigma}^2_{\mathcal{U}_u} \odot \boldsymbol{\epsilon}^{(l)}, \text{with } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}),\end{aligned} \tag{13}$$

where L is size of the sampling, $\bullet\ |_d$ denotes the d-th element of $\bullet$, D is the size of dimension of the latent vectors, and $\odot$ indicates the element-wise product, $\boldsymbol{\epsilon}$ is the auxiliary noise variable, and $\mathcal{N}(0, \mathbf{I})$ is the standard normal distribution. As shown in Eq. 13, by a differentiable transformation of an auxiliary 'noise' variable $\boldsymbol{\epsilon}^{(l)}$ [17], we can reparameterize the latent Gaussian embeddings $q_{\phi_1}(\mathbf{Y} \mid \mathbf{W})$ and $q_{\phi_2}(\mathbf{Z} \mid \mathbf{V})$ into $\mathbf{Y}^{(l)}$ and $\mathbf{Z}^{(l)}$, which are deterministic and can be differentiated efficiently using the backpropagation algorithm.

In order to obtain more accurate user profiling results, our proposed model, i.e., CCEM, also works with constraints which enforce the inferred embeddings of users and words subject to the criteria: for a given question, embeddings of users whose answers receive more votes are closer to the embeddings of the words appearing in these answers, compared to the embeddings of those whose answers receive less votes. We elaborate the approach that adopt constraints into the co-embedding process in the following section.

4.4 Co-embedding with Constraints

In this section, we detail the method of applying additional constraints. Formally, a single constraint can be defined as follows:

$$S(\mathbf{Y}_i, \mathbf{Z}_u) \geq S(\mathbf{Y}_i, \mathbf{Z}_v), \tag{14}$$

which indicates that user u has more votes than user v for word i. Here, $\mathbf{Y}_i$, $\mathbf{Z}_u$ and $\mathbf{Z}_v$ represent the embeddings of word i, user u and user v, respectively. The function $S(\cdot,\cdot)$ is a scoring function that is introduced to compute the similarly between user embeddings and word embeddings (also called similarity computing function in this paper). The specific form of $S(\cdot,\cdot)$ will be discussed later.

In order to satisfy the constraints while inferring the embeddings of users and words, CCEM builds up on the co-embeddings method as we describe above and extends it by using a Max-margin methods to solve the constraints. Max-margin methods, such as SVM-Rank [15], which adopts SVM method to perform pairwise classification, are usually applied into information retrieve. Specifically, given a set of training queries $\{q_i\}_{i=1}^n$, with n being the number of queries, their associate document pairs $(x_u^{(i)}, x_v^{(i)}) \in S_i$, and the corresponding ground truth label $y_{u,v}^{(i)}$, the standard SVM-Rank is computed as below:

$$\begin{aligned}\min_{w,b,\xi}\ & \frac{1}{2}||w||_2^2 + \lambda\sum_{i=1}^{n}\sum_{(u,v)\in S_i}\xi^i_{(u,v)}\\ s.t.\ \ & y_{u,v}^{(i)}[w^T(x_u^{(i)} - x_v^{(i)}) + b] \geq 1 - \xi^i_{(u,v)},\\ & \xi^i_{(u,v)} \geq 0, \quad i = 1,\dots,n,\end{aligned} \tag{15}$$

where $\xi^i_{(u,v)}$ is a slack variable,which represents the tolerance of incorrect ranking document u and document v on query i, and note that a scoring function is used to compute the relevance between query i and document u, i.e., $f(q_i, x_u^i) = w^T x_u^{(i)} + b$.

Inspired by SVM-Rank, in our task, similarly, given all N words, i.e., the size of vocabulary in our task, their associate user pair $(\mathbf{Z}_u, \mathbf{Z}_v) \in \mathcal{S}_{w_i}$, which created by counting the votes of word i on each users, and the corresponding ground truth label $y_{u,v}^{(i)}$, the formula in Eq. 15 can be written as below:

$$\begin{aligned}\min_{w,b,\xi} \mathcal{L}_{SVM} = \min_{w,b,\xi}\ & \frac{1}{2}||w||_2^2 + \lambda\sum_{i=1}^{N}\sum_{(u,v)\in\mathcal{S}_{w_i}}\xi^i_{(u,v)}\\ s.t.\ \ & y_{u,v}^{(i)}[w^T(<\mathbf{Y}_i, \mathbf{Z}_u> - <\mathbf{Y}_i, \mathbf{Z}_v>) + b] \geq 1 - \xi^i_{(u,v)}\\ & \xi^i_{(u,v)} \geq 0, \quad i = 1,\dots,N,\end{aligned} \tag{16}$$

where, similarly, $\xi^i_{(u,v)}$ represents the tolerance of incorrect ranking user u and user v on word i, i.e., the tolerance of the constraints between user u and user v on word i not being satisfied. The scoring function in this part can be seen as the function of computing the similarity between user embeddings and word embeddings. Unlike the standard SVM-Rank, this scoring function receive two inputs, i.e., user embeddings and word embeddings. Note that if one user has high expertise for a specific word, their embeddings should have a high similarity. To this end, we make an element-wise product between them, (e.g., $<\mathbf{Y}_i, \mathbf{Z}_u>$ denote element-wise product between $\mathbf{Y}_i$ and $\mathbf{Z}_u$), and thus the scoring function can be written as: $S(\mathbf{Y}_i, \mathbf{Z}_u) = w^T <\mathbf{Y}_i, \mathbf{Z}_u> +b$. Then we can obtain the parameters of the scoring function $S(\cdot,\cdot)$, i.e., w and b, by solving the quadratic programming problem as discussed above.

To obtain the embeddings of both users and words with additional constraints, CCEM aims to optimize the max-margin classifier of SVM-Rank as well as the ELBO. The final objective for learning is defined as follows:

$$\min_{\theta,\phi,w,b,\xi} \mathcal{L} = \min_{\theta,\phi,w,b,\xi} \mathcal{L}(\theta,\phi;\mathbf{W},\mathbf{V}) + \mathcal{L}_{SVM} \tag{17}$$
$$s.t. \quad y_{a,b}^{(i)}[w^T(<\mathrm{Y}_i,\mathbf{Z}_u> - <\mathrm{Y}_i,\mathbf{Z}_v>) + b] \geq 1 - \xi_{(u,v)}^i$$
$$\xi_{(u,v)}^i \geq 0, \quad i = 1,...,n.$$

5 OPTIMIZATION

The optimization of Eq. 17 involves the training of parameters for the ELBO, i.e. the parameters of the inference model ϕ and the parameters of the generative model θ, and the parameters for the SVM-Rank loss, i.e. the parameters w and b in the similarity computing function. In order to jointly train our model to obtain the optimal parameters, we first fix the ELBO parameters, update and optimize the parameters in the SVM-Rank, and then fix the parameters in SVM-Rank loss, update and optimize the parameters in the ELBO. This enables the variational latent variables subject to the constraints while still being able to co-embed users and words in the same semantic space. The optimization process of CCEM is described in the following subsection.

5.1 Optimize θ and ϕ

When w and b are fixed, we optimize the following objective function:

$$\min_{\theta,\phi} \mathcal{L}(\theta,\phi;\mathbf{W},\mathbf{V}) - \alpha[\sum_{i=1}^{N}\sum_{(u,v)\in C_{w_i}} S(\mathrm{Y}_i,\mathbf{Z}_u) - S(\mathrm{Y}_i,\mathbf{Z}_v)]$$
$$s.t. \quad \alpha \geq 0 \tag{18}$$

where α is a trade-off parameter that governs the contribution of the constraints over the optimization process, $S(\cdot,\cdot)$ is the similarity computing function, i.e. the scoring function, and C_{W_i} is a set of positive users pairs on word i. Each pair $(\mathbf{Z}_u,\mathbf{Z}_v) \in C_{w_i}$ satisfies that the embeddings of word i (i.e. Y_i) has higher similarity with the embeddings of users u (i.e. $\mathbf{Z}_u$) than user v (i.e. $\mathbf{Z}_v$). Note that the constraints part of objective in Eq. 18 can be treated as a regularizer, which enforces the embedding of the users closer to those of the words after the optimization.

In order to optimize the objective in Eq. 18, we apply two neural network models, i.e., the *inference model* q_ϕ with trainable parameters ϕ and the *generative model* p_θ with trainable parameters θ, to perform gradient decent for learning all the model parameters.

Inference Model q_ϕ. In this part, we aim at encoding from observation variables, i.e. words co-occurrence matrix $\mathbf{W}$ and users co-occurrence matrix $\mathbf{V}$, to Gaussian embeddings. We apply two-layer fully connected neural networks for our two encoders, one for inferring the embeddings of the users and another for inferring the embeddings of the words, respectively. Specifically, the network structure for inferring user and word embeddings is defined as follows:

$$\mathbf{H}_{\mathcal{U}}^{(1)} = \text{relu}\left(\mathbf{V}\mathbf{W}_{\mathcal{U}}^{(0)} + \mathbf{b}_{\mathcal{U}}^{(0)}\right),$$
$$[\boldsymbol{\mu}_{\mathcal{U}},\boldsymbol{\sigma}_{\mathcal{U}}^2] = \mathbf{H}_{\mathcal{U}}^{(1)}\mathbf{W}_{\mathcal{U}}^{(1)} + \mathbf{b}_{\mathcal{U}}^{(1)}, \tag{19}$$
$$\mathbf{H}_{\mathcal{W}}^{(1)} = \text{relu}\left(\mathbf{W}\mathbf{W}_{\mathcal{W}}^{(0)} + \mathbf{b}_{\mathcal{W}}^{(0)}\right),$$
$$[\boldsymbol{\mu}_{\mathcal{W}},\boldsymbol{\sigma}_{\mathcal{W}}^2] = \mathbf{H}_{\mathcal{W}}^{(1)}\mathbf{W}_{\mathcal{W}}^{(1)} + \mathbf{b}_{\mathcal{W}}^{(1)}, \tag{20}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the means and variances of the learned embeddings of users and words, $\mathbf{b}$ is the bias and relu$(\cdot)$ is the rectified linear unit activation function [13]. For brevity, we let $\phi = [\phi_1,\phi_2]$ denote all the trainable parameters of encoder with $\phi_1 = [\mathbf{W}_{\mathcal{W}}^{(0)},\mathbf{W}_{\mathcal{W}}^{(1)},\mathbf{b}_{\mathcal{W}}^{(0)},\mathbf{b}_{\mathcal{W}}^{(1)}]$ being the trainable weights for the word inference layers and $\phi_2 = [\mathbf{W}_{\mathcal{U}}^{(0)},\mathbf{W}_{\mathcal{U}}^{(1)},\mathbf{b}_{\mathcal{U}}^{(0)},\mathbf{b}_{\mathcal{U}}^{(1)}]$ being trainable weights for the user inference layers.

After having obtained all the means and variances for all the Gaussian embeddings of words and users, the reparameterization trick is applied to transform from the latent Gaussian random variables to the deterministic values of Y and $\mathbf{Z}$, which are differentiable and capable to propagation the gradient between the inference model and generative model.

Generative Model g_θ. The generative model aims to reconstruct the visible variable, i.e., the word-word co-occurrence matrix $\mathbf{W}$ and the user-word co-occurrence matrix $\mathbf{V}$, according to the deterministic latent embeddings Y and $\mathbf{Z}$. Here we denote the word-word generative model network by g_{θ_1}. Then g_{θ_1} takes latent embeddings of words Y_i and Y_j as input, and outputs the parameters of the corresponding distribution:

$$[\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{W}}(i,j)},\boldsymbol{\sigma}_{\mathcal{E}_{\mathcal{W}}(i,j)}^2] = g_{\theta_1}(\mathrm{Y}_i,\mathrm{Y}_j). \tag{21}$$

The value of a word pair $\mathbf{W}_{ij} \in \mathbf{W}$ can be binary or real, which are treated by different generative distributions:

(i) For real-valued co-occurrence pairs, e.g., the co-occurrence frequency between two words i and j can be generative by:

$$p_{\theta_1}(\mathbf{W}_{ij} \mid \mathrm{Y}_i,\mathrm{Y}_j) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{W}}(i,j)},\boldsymbol{\sigma}_{\mathcal{E}_{\mathcal{W}}(i,j)}^2\mathbf{I}). \tag{22}$$

(ii) For binary-valued co-occurrence pairs, e.g., the probability that word i and word j co-occur in same answers can be:

$$p_{\theta_1}(\mathbf{W}_{ij} \mid \mathrm{Y}_i,\mathrm{Y}_j) = \text{Ber}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{W}}(i,j)}). \tag{23}$$

where $\mathcal{N}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{W}}(i,j)},\boldsymbol{\sigma}_{\mathcal{E}_{\mathcal{W}}(i,j)}^2\mathbf{I})$ and Ber$(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{W}}(i,j)})$ are multivariate Gaussian distribution and Bernoulli distribution parametrized by $\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{W}}(i,j)}$, $\boldsymbol{\sigma}_{\mathcal{E}_{\mathcal{W}}(i,j)}^2\mathbf{I}$ and $\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{W}}(i,j)}$, respectively.

Similarly, we denote the user-word generative model network by g_{θ_2}, which takes latent embeddings of users $\mathbf{Z}_u$ and words Y_i as input, and outputs the parameters of the corresponding distribution:

$$[\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{U}}(u,i)},\boldsymbol{\sigma}_{\mathcal{E}_{\mathcal{U}}(u,i)}^2] = g_{\theta_2}(\mathbf{Z}_u,\mathrm{Y}_i). \tag{24}$$

$\mathbf{V}_{ui}$ also can be binary or real values, and the parameters of $\mathcal{E}_{\mathcal{U}}(u,i)$ corresponding to these two types of value can be generate by the following:

(i) For real-valued co-occurrence pairs, e.g., the probability that word i is associated with user u can be:

$$p_{\theta_1}(\mathbf{V}_{ui} \mid \mathbf{Z}_u,\mathrm{Y}_i) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{U}}(u,i)},\boldsymbol{\sigma}_{\mathcal{E}_{\mathcal{U}}(u,i)}^2\mathbf{I}). \tag{25}$$

(ii) For binary-valued pairs, e.g. the probability whether user u have used word i can be:

$$p_{\theta_1}(\mathbf{V}_{ui} \mid \mathbf{Z}_u,\mathrm{Y}_i) = \text{Ber}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{U}}(u,i)}). \tag{26}$$

We denote $\theta = [\theta_1, \theta_2]$ as all the trainable parameters of generative model. Since all the co-occurrence information including word-word co-occurrence pairs and user-word co-occurrence pairs, are binary-valued, we perform our generative model by inner product between the latent embeddings:

$$g_{\theta_1}(\mathrm{Y}_i, \mathrm{Y}_j) = \mathrm{sigmoid}(\mathrm{Y}_i^{\mathrm{T}}\mathrm{Y}_j),$$
$$g_{\theta_2}(\mathrm{Z}_u, \mathrm{Y}_i) = \mathrm{sigmoid}(\mathrm{Z}_u^{\mathrm{T}}\mathrm{Y}_i), \tag{27}$$

where sigmoid$(\cdot)$ is the sigmoid function.

5.2 Optimize w and b

Once θ and ϕ have been obtained, to optimize the objective in Eq. 17 is equal to optimize SVM-Rank described in §4.4. We can obtain the weight vector w and the bias b, i.e., the parameters of the similarity computing function $S(\cdot, \cdot)$, by solving the quadratic programming problem.

After the optimization is done, we can obtain the embeddings of all users and words. We then obtain the top-K relevant keywords, i.e., $\mathcal{K}_u$, by computing the cosine similarity between the embedding of an input user u and embeddings of each word. Our experimental results show that for user profiling task in question answering communities, the embeddings of users and words obtained by our CCEM model can produce better results than those obtained by several state-of-the-air baselines.

6 EXPERIMENTAL SETUP

6.1 Research Questions

We aim to answer the following research questions that guide the remainder of the paper: (**RQ1**) How is the performance of CCEM on the user profiling compared with the state-of-the-art methods? (**RQ2**) Can the additional constraints of our CCEM improve the performance on the task of user profiling? (**RQ3**) Can the embeddings learned by our proposed co-embedding method effectively measure the semantic similarity between users and words? (**RQ4**) Does the representations inferred by CCEM have good generalization performance? (**RQ5**) How do the trade-off hyper-parameters in optimization process affect the quality of embeddings of both users and words?(**RQ6**) Is CCEM sensitive to the embedding dimensions?

6.2 Dataset

In order to answer our research questions, we conduct experiments on a dataset collected from a well-known Chinese question answering community platform, Zhihu, where questions are posted and answered by its community users. Specifically, we first manually selected users who have expertise in one or multiple fields, such as sport, music, astronomy etc., and then crawled the profile information of these users, all answers they posted and the voting information of these answers.

To obtain each users' relevant words, we first segmented the answers into words by using a tool for Chinese word segment, i.e., the pkuseg tool [37]. Then we removed the stop words and filtered words based on word frequency. By doing so, we can filter out most of the words that have no meanings. Finally, we obtain 2692 users, about 30000 words and corresponding votes on each words.

For effectively evaluate the performance of our proposed model, we employ a number of annotators to manually create the Ground Truth. For each user, an annotator was asked to generate a ranked list of top-k relevant keywords (the number of which was decided by the annotators) that can summarize the user's expertise. In total, 100 annotators took part in the annotation with each of them labelling about 25 users.

6.3 Baselines and Settings

We make comparisons between our proposed CCEM model and the following state-of-the-art algorithms for user profiling:

- **TF-IDF:** It simply uses the TF-IDF algorithm [38] to compute a weight for each word corresponding to each user, and use it to retrieve top-k keywords as profiles for the users.
- **Average Word Embedding (AWE):** It uses the average of all word embeddings corresponding to each user to represent this user. In this way, we use the Chinese word embeddings pre-trained by Directional Skip-Gram model which was applied by Tencent [36].
- **TF-IDF + Word Embedding (TWE):** It first computes the TF-IDF weight of each corresponding word of each user, multiplies each word by its weight, and then computes the average embedding to represent the user. We also use the Chinese word embeddings pre-trained by Tencent.
- **Word2Vec:** This is a well-known word embeddings algorithm proposed by Mikolov et al. [30]. In this baseline model, we regard each user as a special word and the words of answers corresponding to this user as context.
- **Co-embedding Model (CEM):** Co-embedding users and words by using our variational auto-encoder but without the additional constraints.

For convenient comparison, we set the number of dimensions both in CCEM and in the baseline methods, to be 200, since the number of dimensions of the pre-trained Chinese word embeddings generated by DSG [36] is 200. For Word2Vec model, we set the window size to be 5, and at the same time, we also set the window size to 5 for counting co-occurrence information between any two words and build the word-to-word co-occurrence matrix, i.e., **W** as the input of our CCEM and the baseline model, CEM.

6.4 Evaluation Metrics

To evaluate performance, we use standard relevance-oriented evaluation metrics: R-Prec, MAP (Mean Average Precision), Pre@k (Precision at k), NDCG@k (Normalized Discounted Cumulative Gain at k), MRR@k (Mean Reciprocal Rank at k) [8]. R-Prec is the precision after R relevant words have been retrieved, where R is the total number of relevant words for profiling a given user. We also use semantic version of the standard metrics, representing as Pre-S@k, NDCG-S@k, MRR-S@k, respectively. The standard metrics and the corresponding semantic ones differ between them in the way to compute the relevance score of a model output keyword w^* to ground truth keyword w^{gt}. Specifically, for standard metric we let the relevant score be 1 if $w^* = w^{gt}$, otherwise be 0, whereas we let the relevant score be calculated by cosine similarity between the word embeddings of these two keyword w^* and w^{gt} in semantic version. For each metric we let k be 5, 10 and 20 (for MRR and

Table 2: Performance of CCEM and the baselines on MAP, R-Prec, P@5, 10,20, NDCG@5, 10,20, MRR@5,10,20, respectively. The statistical significance is tested using a two-tailed paired t-test. Statistically significant differences between CCEM and the best baseline model are denoted using $^\blacktriangle$at the upper right corner of the CCEM performance scores.

			Pre@			NDCG@			MRR@		
	MAP	R-Prec	5	10	20	5	10	20	1	3	5
TF-IDF	.183	.078	.073	.068	.046	.074	.070	.054	.024	.022	.021
AWE	.196	.111	.081	.069	.050	.087	.076	.061	.032	.022	.031
TWE	.223	.118	.095	.083	.061	.100	.090	.072	.033	.028	.036
Word2Vec	.239	.133	.092	.090	.061	.101	.090	.073	.034	.036	.037
CEM	.275	.154	.141	.130	.097	.143	.135	.110	.045	.046	.051
CCEM	.328$^\blacktriangle$	.194$^\blacktriangle$	.188$^\blacktriangle$	.184$^\blacktriangle$	.121$^\blacktriangle$	.190$^\blacktriangle$	.186$^\blacktriangle$	.141$^\blacktriangle$	.057$^\blacktriangle$	.059$^\blacktriangle$	.067$^\blacktriangle$

Table 3: Profiling performance of CCEM and the baselines on MAP-S, R-Prec-S, P-S@5,10,20, NDCG-S@5,10,20, MRR-S@1,3,5, respectively. Notations for statistically significant differences between CCEM and the best baseline model are the same as those in Table 2.

			Pre-S@			NDCG-S@			MRR-S@		
	MAP-S	R-Prec-S	5	10	20	5	10	20	1	3	5
TF-IDF	.493	.546	.493	.489	.466	.476	.469	.442	.193	.191	.194
AWE	.588	.671	.588	.577	.551	.598	.591	.569	.265	.251	.269
TWE	.593	.683	.601	.591	.563	.622	.613	.588	.268	.257	.273
Word2Vec	.605	.723	.607	.595	.568	.684	.679	.652	.290	.289	.307
CEM	.622	.726	.627	.619	.589	.715	.704	.684	.294	.280	.299
CCEM	.659$^\blacktriangle$	.774$^\blacktriangle$	.647$^\blacktriangle$	.641$^\blacktriangle$	.606$^\blacktriangle$	.757$^\blacktriangle$	.743$^\blacktriangle$	.710$^\blacktriangle$	.336$^\blacktriangle$	.324$^\blacktriangle$	.338$^\blacktriangle$

MRR-S, k being 1, 3 and 5) to compute $M@k$, where M is one of the metrics.

7 RESULTS AND ANALYSIS

In what follows, we discuss and analyze our experimental results and answer the research questions.

7.1 Overall Profiling Performance

(RQ1) We compare the profiling performance of our CCEM with the baselines methods list in 6.3.

Table 2 compares the result of user profiling between CCEM and the baseline methods in terms of standard evaluate metrics that we discussed in 6.4. The ranking of models with respect to the performance is consistent across different evaluation metrics, and in particular this order is observed: CCEM> CEM > Word2Vec ~ TWE > AWE > TFIDF, where note that > denotes the performance difference is statistically significant at a significance level of 95% with Student's two tailed t-test, and ~ denotes the difference is not statistically significant. The standard metrics compute the relevance score between a retrieved keyword w^* to the ground truth keyword w^{gt} in binary manners. We introduce the semantic versions of the standard metrics. Table 3 shows the result. From the table, we can see that our proposed CCEM model significantly outperforms all other baseline models. Specifically, this order can be observed: CCEM> CEM ~ Word2Vec > TWE > AWE > TFIDF.

7.2 Effect of Constraints

(RQ2) We compare the profiling performance of CCEM and CEM, i.e., our CCEM without constraints.

Fig. 3 shows the performance in terms of Pre-S@5, NDCG-S@5, MRR-S@1, MAP-S for the user profiling task by different value of the trade-off parameter α. Note that when α is set to 0, CCEM will ignore constraints and reduce to the baseline CEM model. As can be seen, CCEM significantly outperforms the two baselines on all the metrics, which suggests that our CCEM can efficiently pull users closer to the words they have expertise on while the ELBO can still be efficiently optimized.

7.3 Effect of Co-embedding Users and Words

(RQ3) Now, we examine whether co-embedding users and words into the same semantic space can improve the quality of embeddings. We compare the profiling performance of co-embedding model, i.e., CCEM and CEM, with other best baselines which are not based on co-embedding method.

Table 4 shows the performance of CCEM and other baselines evaluated by the metrics. It can be clearly observed that, co-embedding methods, i.e., CCEM and CEM, significantly outperform the other two methods, i.e., Word2Vec and TWE, which is not co-embedding methods. This demonstrates that our proposed co-embedding method can effectively measure the semantic similarity between the embeddings of users and words, i.e., the embeddings produced by CCEM and CEM significantly contribute to the improvement of the profiling performance.

7.4 Quality of Semantic Representations

(RQ4) We now evaluate the performance of CCEM and the baseline models by perplexity, which is a widely used metric to evaluate the generation of representations [4, 24].

Original perplexity is monotonically decreasing with the likelihood of the documents, and is algebraically equivalent to the inverse of the geometric mean per-word likelihood. In our task of user profiling, to evaluate the quality of both representation of users and words, we modify the way to compute the perplexity as: $Perplexity(\mathcal{K}_u) = \exp(-\frac{\sum_{i=0}^{|\mathcal{K}_u|} \log p(w_{u,i}|u)}{|\mathcal{K}_u|})$, where $\mathcal{K}_u$ denotes a set of top-K keywords corresponding to user u, and $p(w_{u,i}|u) = \pi - arccos(cos(\mathbf{W}_{u,i}, \mathbf{Z}_u))$, with $\mathbf{W}_{u,i}$ and $\mathbf{Z}_u$ representing the embeddings of keyword $w_{u,i}$ and user u, respectively. A lower perplexity score indicates better generalization performance. Fig. 2 shows the mean perplexity performance of CCEM and the baselines models, with different size of top-K keywords set, i.e., different number of K, ranging form 10 to 300. As can be seen, our CCEM consistently outperforms the other baselines models, which demonstrates the good quality of the embeddings of both users and words produced by CCEM.

7.5 Impact of the Trade-off Parameters

(RQ5) Next, we tune the trade-off parameter between the ELBO and constraints in the objective in Eq. 18, i.e., α, and the trade-off parameter between the words and the users, i.e., β, during the optimization process of the co-embedding in Eq. 6, and then evaluate the performance by different values of each trade-off parameters.

When β fixed, we implement CCEM with various values of the parameter α. Specifically, we set the number of α from 0 to 0.003. Fig. 3 shows the performance of our CCEM on different number of

Table 4: Performance comparisons among CCEM, CEM, i.e., CCEM without constraints, and other best baselines on standard metrics and their semantic versions. Notations for statistically significant differences between CCEM and the best baseline model are the same as those in Table 2.

					Pre@			NDCG@			MRR@			Pre-S@			NDCG-S@			MRR-S@		
	MAP	R-Prec	MAP-S	R-Prec-S	5	10	20	5	10	20	1	3	5	5	10	20	5	10	20	1	3	5
TWE	.223	.118	.593	.683	.095	.083	.061	.100	.090	.072	.033	.028	.036	.601	.591	.563	.622	.613	.588	.268	.257	.273
Word2Vec	.223	.118	.605	.723	.092	.090	.061	.101	.090	.073	.034	.036	.037	.607	.595	.568	.684	.679	.652	.290	.289	.307
CEM	.275	.154	.622	.726	.141	.130	.097	.143	.135	.110	.045	.046	.051	.627	.619	.589	.715	.704	.684	.294	.280	.299
CCEM	.328▲	.194▲	.659▲	.774▲	.188▲	.184▲	.121▲	.190▲	.186▲	.141▲	.057▲	.059▲	.067▲	.647▲	.641▲	.606▲	.757▲	.743▲	.710▲	.336▲	.324▲	.338▲

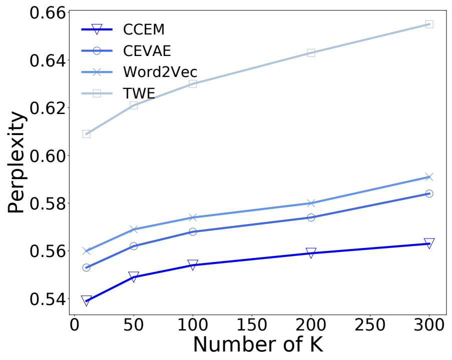
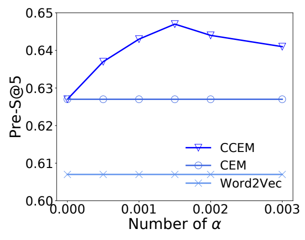
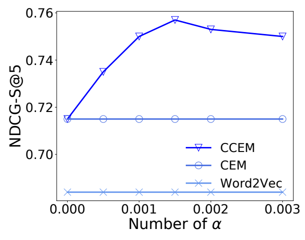


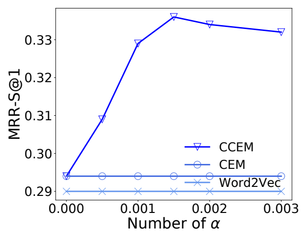
Figure 2: Pre-S@5 and NDCG-S@5 performance of CCEM and the baselines with various sizes of dimensions of the inferred embeddings.



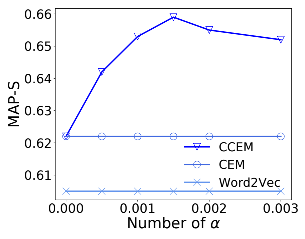
(a) Pre-S@5



(b) NDCG-S@5



(c) MRR-S@1



(d) MAP-S

Figure 3: User profiling performance comparisons among our CCEM with different α and other two best baselines evaluated by Pre-S@5, NDCG-S@5, MRR-S@1 and MAP-S.

α using Precision-S@5, NDCG-S@5, R-Precision-S@5 and MRR-S@1 as representative metrics. As can be seen, the performance evaluated by all these metrics own an upward trend when α is increased from 0 to ~0.0015. An increase of α within this range allows constraint information to be more effectively captured. Another finding is that the performance evaluated by the metrics suffers a slight decrease when α is larger than ~0.0015, since when the number of α is too large, it ignores the reconstruction of embeddings of words and users during the optimization process.

We now let α be fixed, and vary the values of the parameter β and observe the performance of CCEM and CEM. Fig. 4 shows the performance. It is clear from the figure that the performance increases both in CCEM and the baselines with β increasing from 1 to ~1.6. The performance of CCEM and the baseline models suffer a slight decrease when β becomes larger than ~1.6, which demonstrates that paying more attention on the optimization of users would result in better performance for the task of user profiling.

In general, as it can be observed that CCEM consistently performs better than the rest of the models with any α and β values, which demonstrates the robust and superiority of our co-embedding model with additional constraints, i.e., our proposed CCEM model.

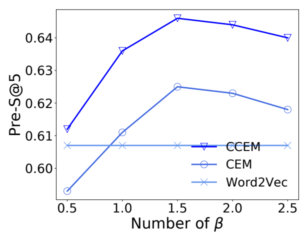
7.6 Dimensions of the Embeddings

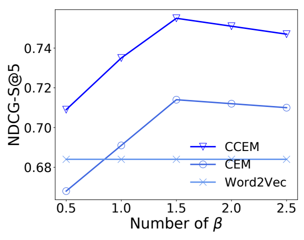
(RQ6) Finally, we tune the sizes of dimensions of the embeddings of both users and words in the models and evaluate their performance.

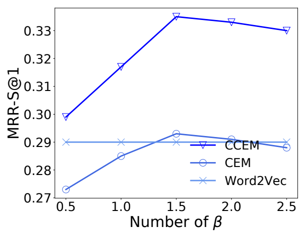
Fig. 5 shows the performance of our CCEM and the best baselines, i.e. CEM and Word2Vec, on different sizes of dimensions varying from 50 to 400. Note that we only report the performance evaluated by Precision-S@5 and NDCG-S@5, since we can also observe the same pattern in terms of other metrics. As can be seen from the figure, the performance owns an upward trend with the number of dimensions both in CCEM and the baselines increasing from 50 to ~250. In addition, the performance of all the models seems to reach a plateau when the dimensions go from ~250 to 400. We can observe that at all different sizes of dimensions, CCEM keeps outperforming all other baselines. All of these findings demonstrate that CCEM is not sensitive to the size of dimensions of the embeddings when the size is set to be large enough, and it is able to consistently improve user profiling performance with various sizes of the dimensions over the best embedding models.

8 CONCLUSION

We have studied the problem of user profiling in Question Answering Communities. To tackle the problem, we have proposed a Constrained Co-embedding Model, called CCEM. Our CCEM is the first attempt to learn the representations of both users and words with additional constraints in the same semantic space. CCEM adopts a variational auto-encoder model with modified network structure which allows the embeddings of users and words to be leaned in the same semantic space. Furthermore, additional constraints are


(a) Pre-S@5


(b) NDCG-S@5


(c) MRR-S@1

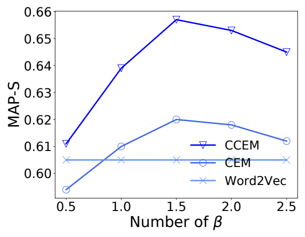
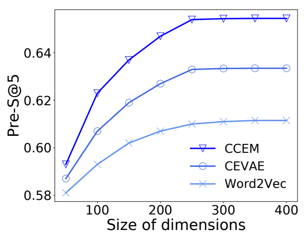

(d) MAP-S

Figure 4: User profiling performance comparisons among CCEM and CEM with different β and the other baselines evaluated by Pre-S@5, NDCG-S@5, MRR-S@1 and MAP-S.


(a) Pre-S@5

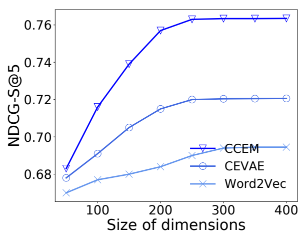

(b) NDCG-S@5

Figure 5: Pre-S@5 and NDCG-S@5 performance of CCEM and the baselines with various sizes of dimensions of the inferred embeddings.

applied in the co-embedding inference that enforce the embeddings of users and words are subject to specific criteria, leading to a more accurate user profile performance. Experimental results on Zhihu dataset demonstrate the effectiveness of the proposed model.

As to future work, we intend to apply our inferred embeddings to boost the performance of other related applications, e.g., answer predictions in question answering communities. We also plan to utilize other auxiliary information, e.g., users’ social circles, to further help to infer the embeddings in the communities.

Acknowledgments. This research was partially supported by the National Natural Science Foundation of China (Grant No. 61906219).

REFERENCES

[1] H. Bai and H. Zhao. Deep enhanced representation for implicit discourse relation recognition. *proceedings of COLING*, 2018.
[2] K. Balog, T. Bogers, L. Azzopardi, M. De Rijke, and A. Van Den Bosch. Broad expertise retrieval in sparse data environments. In *Proceedings of SIGIR*, pages 551–558. ACM, 2007.
[3] K. Balog, M. De Rijke, et al. Determining expert profiles (with an application to expert finding). In *IJCAI*, volume 7, pages 2657–2662, 2007.
[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3(Jan): 993–1022, 2003.
[5] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *proceedings of CONLL*, 2015.
[6] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang. Representation learning for attributed multiplex heterogeneous network. *Proceedings of SIGKDD*, 2019.
[7] N. Craswell, A. P. de Vries, and I. Soboroff. Overview of the trec 2005 enterprise track. In *Trec*, volume 5, pages 1–7, 2005.
[8] W. B. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading, 2010.
[9] M. De Rijke, K. Balog, T. Bogers, and A. Van Den Bosch. On the evaluation of entity profiles. In *CLEF*, pages 94–99. Springer, 2010.
[10] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in NIPS*, pages 658–666, 2016.
[11] Y. Fang and A. Godavarthy. Modeling the dynamics of personal expertise. In *Proceedings of SIGIR*, pages 1107–1110. ACM, 2014.
[12] A. L. Ginsca and A. Popescu. User profiling for answer quality assessment in q&a communities. In *Proceedings of the 2013 workshop on Data-driven user behavioral modelling and mining from social media*, pages 25–28. ACM, 2013.
[13] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of ASC*, pages 315–323, 2011.
[14] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of SIGKDD*, pages 855–864. ACM, 2016.
[15] R. Herbrich. Large margin rank boundaries for ordinal regression. *Advances in large margin classifiers*, pages 115–132, 2000.
[16] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.
[17] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *proceedings of ICLR*, 2013.
[18] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in NIPS*, pages 3581–3589, 2014.
[19] Y.-Y. Lai, J. Neville, and D. Goldwasser. Transconv: Relationship embedding in social networks. 2019.
[20] R. Lebret and R. Collobert. Word emdeddings through hellinger pca. *proceedings of EACL*, 2013.
[21] S. Liang. Dynamic user profiling for streams of short texts. In *AAAI*, 2018.
[22] S. Liang. Collaborative, dynamic and diversified user profiling. In *AAAI*, 2019.
[23] S. Liang and M. de Rijke. Formal language models for finding groups of experts. *Information Processing & Management*, 52(4):529–549, 2016.
[24] S. Liang, E. Yilmaz, and E. Kanoulas. Dynamic clustering of streaming short documents. In *Proceedings of SIGKDD*, pages 995–1004. ACM, 2016.
[25] S. Liang, X. Zhang, Z. Ren, and E. Kanoulas. Dynamic embeddings for user profiling in twitter. In *Proceedings of SIGKDD*, pages 1764–1773, 2018.
[26] S. Liang, E. Yilmaz, and E. Kanoulas. Collaboratively tracking interests for user clustering in streams of short texts. *IEEE Transactions on Knowledge and Data Engineering*, 31(2):257–272, 2019.
[27] Z. Meng, S. Liang, H. Bao, and X. Zhang. Co-embedding attributed networks. In *Proceedings of WSDM*, pages 393–401, 2019.
[28] Y. Miao, L. Yu, and P. Blunsom. Neural variational inference for text processing. In *ICML*, pages 1727–1736, 2016.
[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *proceedings of ICLR*, 2013.
[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*, pages 3111–3119, 2013.
[31] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543, 2014.
[32] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *proceedings of NAACL*, 2018.
[33] J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, and J. Tang. Netsmf: Large-scale network embedding as sparse matrix factorization. In *Proceedings of WWW*, 2019.
[34] F. Riahi, Z. Zolaktaf, M. Shafiei, and E. Milios. Finding expert users in community question answering. In *Proceedings of WWW*, pages 791–798. ACM, 2012.
[35] J. Rybak, K. Balog, and K. Nørvåg. Temporal expertise profiling. In *ECIR*, pages 540–546. Springer, 2014.
[36] Y. Song, S. Shi, J. Li, and H. Zhang. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of NAACL*, pages 175–180, 2018.
[37] X. Sun, H. Wang, and W. Li. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *Proceedings of ACL*, pages 253–262, 2012.
[38] J. F. Wiley. *R Deep Learning Essentials*. Packt Publishing Ltd, 2016.
[39] Z.-M. Zhou, M. Lan, Z.-Y. Niu, and Y. Lu. Exploiting user profile information for answer ranking in cqa. In *Proceedings of WWW*, pages 767–774. ACM, 2012.