Profiling Users for Question Answering Communities via Flow-based Constrained Co-embedding Model

SHANGSONG LIANG, Sun Yat-sen University
YUPENG LUO, Sun Yat-sen University
ZAIQIAO MENG, University of Glasgow

In this paper, we study the task of user profiling in question answering communities (QACs). Previous user profiling algorithms suffer from a number of defects: they regard users and words as atomic units, leading to the mismatch between them; they are designed for other applications but not for QACs; some semantic profiling algorithms do not co-embed users and words, leading to the affinity measurement between them difficult. To improve the profiling performance, we propose a neural Flow-based Constrained Co-embedding Model, abbreviated as FCCM. FCCM jointly co-embeds the vector representations of both users and words in QACs such that the affinities between them can be semantically measured. Specifically, FCCM extends the standard variational auto-encoder model to enforce the inferred embeddings of users and words subject to the voting constraint, i.e., given a question and the users who answer this question in the community, representations of the users whose answers receive more votes are closer to the representations of the words associated with these answers, compared with representations of whose receiving fewer votes. In addition, FCCM integrates normalizing flow into the variational auto-encoder framework to avoid the assumption that the distributions of the embeddings are Gaussian, making the inferred embeddings fit the real distributions of the data better. Experimental results on a Chinese Zhihu question answering dataset demonstrate the effectiveness of our proposed FCCM model for the task of user profiling in QACs.

CCS Concepts: • **Information systems** → **Retrieval models and ranking**;

Additional Key Words and Phrases: Co-embedding, Variational Auto-encoder, User Profiling, Question Answering Communities

ACM Reference format:
Shangsong Liang, Yupeng Luo, and Zaiqiao Meng. 2021. Profiling Users for Question Answering Communities via Flow-based Constrained Co-embedding Model. *ACM Transactions on Information Systems* 1, 1, Article 1 (December 2021), 38 pages.
DOI: 0000001.0000001

1 INTRODUCTION

Following the popularity of web 2.0 technologies, there has been a boom in a broad spectrum of knowledge sharing across a variety of communities. One of such communities is the Question Answering Community (QAC), which imparts and transfers experience and knowledge in the form

Shangsong Liang is with the Sun Yat-sen University, Guangzhou, 510006, China, and the Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, 7909, United Arab Emirates; email: liangshangsong@gmail.com;
Yupeng Luo is with the Sun Yat-sen University, Guangzhou, 510006, China; email: luoyp21@gmail.com;
Zaiqiao Meng (corresponding author) is with the University of Glasgow, Glasgow, G12 8QQ, United Kingdom; email: zaiqiao.meng@gmail.com;
 1046-8188/2021/12-ART1 $15.00
DOI: 0000001.0000001

of questions and answers. Some QACs are online platforms for general discussions (such as Zhihu [1] and Quora [2], where people discuss and share their daily activities and personal opinions), while others are online platforms specialized for technical knowledge translations (such as Stack Overflow where people discuss and share their programming skills). QACs enable users to post questions, and provide answers to questions where they have expertise on the topics of these questions; and in particular, some of them allow users to turn the thumbs up on the answers provided by others if they agree, and turn the thumbs down if they disagree. The goal of QACs is to provide relevant, high-quality and instantaneous answers in response to questions posted by the users in QACs. The key to achieve this goal entails the access of users' profiles such that those who are knowledgeable on the topics of the questions can be effectively identified and recommended to answer the questions either manually or automatically. Accordingly, in this paper, *we aim at addressing the user profiling task in QACs: given a set of users, the questions asked by them, the answers associated with the questions, and the auxiliary information such as thumbs-up and thumbs-down on the answers in QACs, identify a short rank list of keywords to profile users' expertise.*

The task of user profiling has been widely studied since the launch of the expert finding task in enterprise corpora in the Enterprise track of TREC 2005(Text REtrieval Conference) [8]. Language models [1, 59] are utilized to extract keywords as users' profiles from documents in enterprise corpora. Probabilistic models [17] achieve the goal of expert profiling by further developing several probabilistic techniques to improve the estimation of language models, including incorporating topic expansion using a mixture model to put different weights on the matching of an expert, and defining a candidate prior for topic generation models. Temporal hierarchical expertise profiling approach [65] is proposed to further construct temporal profiles of authors in academic databases based on documents labelled with leaf-level categories from a topic taxonomy. Organizational hierarchies and content of documents are taken into account to generate words for members' profiles in organization [31]. In [30], a user's domain knowledge for a specific category is represented as a user knowledge profile derived from the content and quality measures of the user's historical posted documents in that category. In [39], a weakly supervised approach is proposed for user profile extraction from Twitter, where users' profiles are used as a distant source of supervision for extraction of their attributes from user-generated texts. A user profiling model is proposed in [40], where users' attributes and social communities of their friends are jointly profiled by capturing the correlations between users' attributes and their communities. A probabilistic dynamic model [18] is proposed to recognize how users update their expertise in academic community by profiling users' dynamic experience and expertise. In the e-commerce recommender systems domain, a hierarchical user profiling model [23] is proposed to precisely model users' diverse interests during recommendation. To automatically infer users' profiles from their digital footprint in social media, a Markov random field model [19] with hinge loss is proposed to infer a variety of user characteristics, such as, age, gender and personality traits, which can then be compiled to generate user profiles.

Although previous expert profiling models, e.g., those aforementioned, may be directly applied to address the task of user profiling in the context of QACs, they suffer from the following defects: (1) They regard entities such as users and words as atomic units, which would lead to the poor performance of user profiling, as the mismatches between users and words that are used for profiling may occur. (2) They are designed for a number of different applications or other platforms but not for QACs, resulting in how to utilize information specific to QACs such as thumbs-up and thumbs-down of an answer, and the connections between users and answers, to enhance the user profiling performance is still unknown. (3) Users and words are modelled in two different

[1] https://www.zhihu.com
[2] https://www.quora.com

disjoint spaces, resulting in the semantic distances between them are hard to measure. (4) Most of them address the user profiling task by traditional techniques such as language models, but neural network techniques have shown their superior in a spectrum of applications such as text matching [32] and social graph representations [51] . Thus whether neural network techniques can boost the performance of user profiling in QACs is of great value to investigate.

To mitigate the defects of the aforementioned user profiling algorithms, we propose a **F**low-based **C**onstrained **C**o-embedding **M**odel (**FCCM**) that aims at addressing the user profiling task in QACs. Unlike most of the previous work that naively models entities, e.g., words and users, as unique atomic units, our proposed FCCM model deals with user profiling task in QACs by representing users and words as vectors in the same semantic space such that their semantic affinities can be effectively measured. In particular, FCCM aims to jointly infer semantic representations of two different categories of entities, users and words, in the same semantic space via neural Variational Auto-Encoders (VAEs) [36], the goal of which is to semantically capture the affinities between them for the task of user profiling in QACs. To obtain better representations of users and words for user profiling in QACs, FCCM not only utilizes traditional information such as questions and answers posted by the users but also fully utilizes the unique auxiliary information available in QACs, i.e., the thumbs-up and thumbs-down of answers. Specifically, it enforces the embeddings of users and words subject to the *voting constraint* during the inference of the embeddings of users and words: given a question in QACs, the semantic representations of users whose answers receive more votes from other users will be enforced to be closer to the semantic representations of the words appearing in the answers than the representations of words appearing in other answers that received fewer votes. To achieve this, our FCCM extends the standard variational auto-encoders with additional constrains to obtain better embeddings of users and words. Previous VAEs based embedding methods usually take the Gaussian distribution as the initial variational posterior due to the mathematical convenience, but it has been shown that modelling the posterior as Gaussian distribution is not always reasonable in many real-world applications and leads to unstable results and poor performance [25, 62]. Due to the heterogeneity between users and words, the typical uniform Gaussian assumption of these two posteriors in the variational auto-encoders of our FCCM model leads to poor posterior approximations. To obtain better posterior approximations, i.e., to effectively obtain the embeddings of both the users and words, we integrate a normalizing flow [62] into our FCCM. It transforms an initial variational posterior, which is usually with Gaussian distribution or other relatively simple distribution with known and cheaply computable probability density function, to a more flexible posterior by applying a sequence of invertible and smooth mappings.

To evaluate the effectiveness of our proposed model, we conduct experiments on a real QAC dataset. The experimental result shows that our proposed co-embedding model can effectively measure the semantic affinities between the embedding representations of users and words, and adding voting constraint has significant performance boost, both of which help to improve the user profiling performance in QACs and make our model perform significantly better than the baseline models.

The main contributions of the paper are summarized as the followings:

(1) A flow-based constrained co-embedding model, FCCM, is proposed to jointly learn the vector representations of users and words in the same semantic space, which can then be utilized to measure the semantic affinities between users and words for user profiling in QACs.
(2) Our FCCM, extending the standard variational auto-encoder algorithm to address the task of user profiling in QACs, consists of a neural encoder that maps the input entities into

flexible distributions and a neural decoder that is able to effectively reconstruct the input entities based on the semantic representations of the input entities.
(3) Unique auxiliary information in QACs, in particular, the thumbs-up and thumbs-down of answers, is taken into account as voting constrains in the proposed FCCM model such that the inferred vector representations of users and words can be subject to specific criteria that helps to improve the inferred representations for user profiling in QACs.
(4) Our FCCM integrates normalizing flow to transform from an initial variational posterior, which is usually with Gaussian distribution or other relatively simple distribution with known and cheaply computable probability density function, to a more flexible posterior by applying a sequence of invertible and smooth mappings.
(5) We propose an effective inference algorithm to obtain the optimal parameters in our FCCM model. Our inference algorithm is able to iteratively update the objective to obtain the optimal parameters such that the variational latent variable subject to the voting constrains while still being able to co-embed users and words in the same semantic representation space.
(6) Through our proposed FCCM model to address the task of user profiling in QACs as an example, we would provide inspiration and insight on how to extend standard VAE to address other tasks in information retrieval, where semantic vector representations of multiple categories of entities need to be inferred in the same space.
(7) Comprehensive experiments are conducted on the QAC dataset that is crawled from a Chinese QAC, Zhihu, the most well-known QAC in China. Experimental results illustrate that our proposed FCCM model is able to yield the best performance for user profiling task in QAC, compared with the state-of-the-art baselines.

This paper is an extended version of our previous paper [47], where we proposed a constrained co-embedding model (CCEM). The main extension of this paper is threefold: (1) We propose a flow-based constrained co-embedding model (FCCM) by integrating normalizing flow into CCEM, which can avoid the assumption that the distributions of the embeddings are Gaussian, making the inferred embeddings fit the real distributions of the data better. (2) We provide and discuss more recent related work and detailed information of our proposed FCCM model in this extension. (3) This extension details the derivations of the Evidence Lower BOund (ELBO) of the co-embedding model, the logarithm of variational posterior, and the final ELBO with the normalizing flow. (4) Detailed optimization of the proposed model FCCM is also provided. (5) We made comparisons between the performance of our FCCM and that of more baselines and conducted additional experiments to demonstrate the effectiveness of our new model, i.e., FCCM.

The remainder of the paper can be organized as: We first introduce the related work in Section §2 and explain the notation and task formulation in Section §3. Then, we detail our proposed model FCCM in Section §4 and describe its optimization process in Section §5. Section §6 describes the experiment setup. Section §7 analyses the experimental results. Finally, we conclude the paper in Section §8.

2 RELATED WORK

There are three lines of related work, namely, user profiling, variational auto-encoders and embedding models. In the following, we only discuss the most related models and algorithms.

2.1 User Profiling

User profiling, also called expert profiling, is one of the core tasks in the field of information retrieval [2, 8]. The task has been gaining significant attention since the launch of the expert finding

task of the TREC 2005 enterprise track [8, 43], the goal of which is to generate a set of keywords that is able to describe the expertise of a given expert. Balog et al. [1, 2] propose a generative language modelling approach to address the user profiling task based on a heterogeneous corpus made up of a large organization's intranet, and it is the earliest study of user profiling. Later, Fang and Godavarthy [18] address the problem of dynamic user profiling by a probabilistic model that characterizes the dynamics of personal expertise, where authors' academic publications are used to learn the personal research interests. The Author-Topic models are widely used for the task of user profiling. Specifically, these kinds of models are usually based on Latent Dirichlet Allocation (LDA) algorithm, which represents each document in a collection of documents in the form of a probability distribution. The first Author-Topic model is proposed by Rosen-Zvi et al. [64], which not only identifies topics of documents but also profiles users' expertise. Since then, some work related to Author-Topic model has emerged. For instance, Daud [10] propose a Temporal-Author-Topic model that can model the topic distribution of an author over time. Jiang et al. [28] propose a model that integrates Author-Topic model and collaborative filtering. In addition, user information such as textual descriptions of photos is also used in their work. An evaluation method is proposed in [12] for user profiling, which allows for weighted non-exact matches between system-produced and ground-truth keywords. Liang [41] propose a model to profile social users in the context of streaming short documents by using the dynamic topic model. A user profiling algorithm is proposed by Liang [42], which consists of two models: collaborative interest tracking topic model (CITM) and streaming keyword diversification model (SKDM). Liang et al. [46] propose a model to tackle the user profiling task in Twitter by using topic model and word embedding approach. The Heterogeneous Graph Attention Networks have also been utilized to address the semi-supervised user profiling task [6]; however, it cannot be directly applied to address the user profiling task in our QAC scenario, where the voting information should be infused and the profiling keywords need to be inferred in an unsupervised manner. More recently, Gu et al. [23] formulate a novel hierarchical user profiling system, which aims to precisely model users' diverse interests in E-commerce recommender systems. Farnadi et al. [19] propose a mechanism to infer a variety of user characteristics, such as age, gender and personality traits, which can then be compiled into a user profile. Some studies on dynamic user profiling have also received extensive attention. All these user profiling algorithms are designed for enterprise search [1] and academic web service platforms [18, 45] but not for community question answering. In community question answering platforms, exact profiling for users can boost the performance of answer ranking, expert finding, and tag recommendation. However, 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) [36] have gained their popularity in a spectrum of applications such as item recommendation [7] and conversation generation [74] due to their performance superior over many traditional techniques and their high interpretability. Standard VAEs inherit auto-encoder architecture, but make strong assumptions with regard to the distributions of latent variables that are assumed to be Gaussian. VAEs use variational inference approach to learn latent representations for observed entities by optimizing a lower bound of the log evidence likelihood, where the reparameterization trick [36] and Stochastic Gradient Variational Bayes (SGVB) estimator are utilized for an end-to-end training. VAEs can be trained by two types of neural networks: an inference network that maps observed variables into latent representations, and a generative network that reconstructs the latent representations to the corresponding observed variables. Many variants of VAEs have been proposed to tackle problems in a variety of fields: Kingma et al. [34]

propose to tackle the problem of semi-supervised classification by adopting a deep generative model with variational inference. A modified version of variational autoencoder [16] is proposed to generate realistic and high-resolution images. Bowman et al. [5] propose to generate well-formed sentences from a continuous space by adopting an RNN-based variational autoencoder that incorporates distributed latent representations of entire sentences. A generic variational inference framework [52] is proposed for generative and conditional models on textual data. It constructs an inference network conditioned on the discrete text input to provide the variational distribution. Meng et al. [49] modify the network structure of the standard variational autoencoder, such that it can map nodes and attributes of a network into the same semantic embedding space, and even can be applied into the semi-supervised scenario [50]. Davidson et al. [11] propose a Hyperspherical VAE model, which can map the input entities into the corresponding latent von Mises-Fisher (vMF) distributions. More recently, based on VAEs framework, Dieng et al. [14] propose a topic model that extends LDA, which can learn interpretable topic representation when working with large and heavy-tailed vocabularies.

However, VAEs that work with standard variational auto-encoder assume the variational posterior to be Gaussian, which may not be held in many real-world datasets, resulting in poor performance. To tackle this problem, normalizing flow [62] is integrated into the standard VAEs framework. A normalizing flow in VAE works with a transformation of a probability density through a sequence of invertible mappings, such that the initial variational posterior can effectively approximate the true posterior. The key to the normalizing flow is the appropriate choice of functions in flow, which is able to yield simple computation of the determinant of the Jacobian matrix. There are several appropriate choices of functions [15, 20, 25, 33, 35, 55, 60, 69]. Kingma et al. [35] propose the Inverse Autoregressive Flow (IAF), which makes the Jacobian matrix a triangle one, resulting in the corresponding determinant being just the product of the main-diagonal elements. Dinh et al. [15] propose an unsupervised learning algorithm named Real NVP, which uses the real-valued non-volume preserving (real NVP) transformations. Papamakarios et al. [55] propose the Masked Autoregressive Flow (MAF), which can evaluate and train on parallel computing architectures faster by using Masked Auto-encoder for Distribution Estimation (MADE) [20] as a building block. Kingma and Dhariwal [33] propose a new type of generative flow named Glow and demonstrate improved quantitative performance in terms of log-likelihood on standard image modelling benchmarks.

In our 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 votes (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 Embedding Models

There are two categories of embedding models related to our work: word embedding models and network embedding models.

Word embedding is a task that trains 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 [54] had been proposed, which leverages local data, i.e., words' context, to learn syntactic and semantic relationships between words. Specifically, word2vec is a shallow neural network model, and can be implemented by two different networks, i.e., the Continuous Bag of Words (CBOW) and Skip-Gram [54]. The goal of CBOW is to predict the generation probability of the current word based on the words that appear in the context, whereas Skip-gram predicts the context based on the current word. Some other extended work, such as those

proposed in [29, 46, 53], build their models based on the Skip-gram framework. Besides, other ways of generating embeddings have surfaced, which leverage global information to arrive at vector representations for words. Glove model [56] first builds a global co-occurrence matrix that counts the co-occurrence frequency between any two words by setting a sliding window with a fixed size, then trains the model only on the non-zero elements in this co-occurrence matrix, rather than on the entire sparse matrix or specific context. Lebret and Collobert [38] propose a word embedding model that also first utilizes a word-to-word co-occurrence matrix, and then simplifies the word embeddings computation through a Hellinger Principal Component Analysis (Hellinger PCA) of the word co-occurrence matrix. More recently, the problem of polysemy of embedding has gained significant interests. Peters et al. [58] utilize two BiLSTM models to capture the information of context, such that the obtained word embedding can vary across linguistic contexts. Radford et al. [61] propose a pre-train model that trained by utilizing language model that constructed through Multilayer and single direction Transformer encoder [71]. Similar to the work done by Radford et al. [61], Devlin et al. [13] also propose a pre-train model that uses language models for training. It not only uses text information in one single direction but uses information in both directions of the input text.

Network embedding is a task that represents each node in a network to low-dimensional vector and effectively preserves the network structure information as well. Existing network embedding models can be classified into three categories: (1) Structure-preserving network embedding models, such as DeepWalk [57], Node2vec [22], LINE [68] and MMDW [70]; (2) Network embedding models with side information include, e.g., GCN [37], CAN [49] and LANE [27]; (3) Network embedding models with supervise learning models, which are used to solve some specific analytic tasks, such as network alignment [48] and anomaly detection [26]. To preserve the structure information of a network, Deepwalk discovers that the distribution of nodes appearing in short random walks is similar to the that of words in natural language so that it utilizes the Skip-Gram with negative sampling neural network architecture originally proposed for word embedding. Node2vec [22] utilizes both breadth-first sampling (BFS) and depth-first sampling (DFS) to sample the neighbourhood nodes. LINE [68] is a network embedding model that is proposed for serving large scale network structure, and it considers to preserve the first and second order proximities. MMDW [70] integrates Deepwalk and Max-margin method, i.e., support vector machine (SVM), and by optimizing both the max-margin classifier of SVM and matrix factorization based on DeepWalk, the obtained node embedding can have more discriminative ability. To capture various side information in a network, Huang et al. [27] proposed the LANE model that utilizes the labelling information of nodes based on matrix factorization. LANE constructs the corresponding affinity matrices of the node attributes, network structure, and labels by utilizing the cosine similarity to calculate. The Co-embedding Attributed Network (CAN) [49] is an attributed network embedding model that is built based on the GCN [37] model and variational autoencoder, which archives the state-of-the-art performance in both the link prediction task and the attribute inference task. The HAN (Heterogeneous graph Attention Network) model [72] is a semi-supervised graph neural network which employs node-level attention and semantic-level attention simultaneously. In addition, network embedding models are also utilized for solving some specific analytic tasks, for example, Man et al. [48] propose a network embedding model to predict the anchor links across social networks. Hu et al. [26] propose a network embedding model to tackle the problem of anomaly detection, where an embedding-based metric is proposed to indicate the anomalousness level of a node. The larger the value of the measure, the higher the propensity for a node being an anomaly node.

Most of these existing embedding methods infer either the word embeddings or node embeddings only, but not infer the embeddings of two different categories of entities (in our setting, users and words) at the same time. Some co-embedding and heterogeneous graph neural models, such as CAN [49] and HAN [72], are able to jointly learning embeddings for multiple entities, but they are designed to address some graph-based tasks, such as link prediction, node classification or node clustering, rather than the user profiling task. Note that although our proposed model is inspired by the CAN [49] model, 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 THE USER PROFILING TASK

In this section, we start by introducing the main notations used across the paper and then provide the formal definition of the user profiling task for QACs that we would like to deal with.

3.1 Notations

To make the notations clear, we follow many previous work to denote our notations used in the paper. Accordingly, we let normal letters, and calligraphy typeface letters denote variables and sets, respectively. E.g., we let F denote the number of users, and $\mathcal{U}$ denote a set of users in our QAC settings. Let bold and uppercase letter denote matrices. E.g., $\mathbf{V}$ denote the user-to-word co-occurrence matrix. Let bold letter with a subscript i denote the i-th row of the matrix, i.e., a vector in the matrix. E.g., $\mathbf{V}_i$ denotes the i-th row of $\mathbf{V}$. Let a vector with a subscript denote a scalar element of the vector. E.g., $\mathbf{V}_{ij}$ denotes a scalar element of the j-th element of $\mathbf{V}_i$. We summarize the main notations used throughout the paper in Table 1.

We let $\mathcal{U}$ and $\mathcal{W}$ be the user set and word set respectively. The answer sets generated by the users are denoted by $\mathcal{A}_\mathcal{U}$, where $\mathcal{A}_\mathcal{U} = \{\mathcal{A}_{u_1}, \mathcal{A}_{u_2}, \ldots, \mathcal{A}_{u_F}\}$ with $\mathcal{A}_u$ being a set of answers generated by user u in response to the questions he/she answered, and F being the total number of users. Voting information is an important indicator showing which answer has higher quality and can solve the questions of a lot of viewers, which we use $\mathcal{T}_{\mathcal{A}_\mathcal{U}} = \{t_{a_1}, t_{a_2}, \ldots, t_{|\mathcal{T}_{\mathcal{A}_\mathcal{U}}|}\}$ to denote the voting information set, with t_a being the number of the accumulated votes for the answer a. Note that in some QACs, such as Zhihu, all the questions are visible for the public so that every users can answer any questions listed in the platform, but most of the questions are set to be anonymous due to the questioners' privacy consideration. However, most of the questions are randomly listed in the platform, hence choosing proper users (or finding related experts) to answer a potential valuable question is crucial for these platforms to maintain their popularities.

3.2 The User Profiling Task

The user profiling task (also called the expert profiling task) in the context of QACs that we study in this paper can be formulated as this: *given a set of users in QACs, the questions asked by them, the answers associated with the questions, and the auxiliary information such as number of thumbs-up (the number of thumbs-up minus the number of thumbs-down) on the answers, infer the semantic representations of two different categories of entities, i.e., users and words, such that a set of keywords are identified as the profiling results for the users' expertise.* Formally, this user profiling task in QACs is to learn a mapping 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 $\mathcal{K}_\mathcal{U} = \{\mathcal{K}_{u_1}, \mathcal{K}_{u_2}, \ldots, \mathcal{K}_{u_F}\}$ 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 describing user u's expertise.

Table 1. The main notations used throughout the paper.

Notation	Description
$\mathcal{W}$	a set of unique words in the vocabulary
$\mathcal{U}$	a set of users in a QAC
$\mathcal{E}_{\mathcal{W}}$	set of word-to-word pairs
$\mathcal{E}_{\mathcal{U}}$	set of user-to-word pairs
$N = \lvert\mathcal{W}\rvert$	the total number of unique words
$F = \lvert\mathcal{U}\rvert$	the total number of unique users
D	size of the dimension of latent variables/representations
$\mathbf{W} \in \mathbb{R}^{N\times N}$	word-to-word co-occurrence matrix
$\mathbf{V} \in \mathbb{R}^{F\times N}$	user-to-word co-occurrence matrix
$\mathbf{Y}$	representation matrix for all the unique words
$\mathbf{Z}$	representation matrix for all the unique users
$\mathbf{R}$	either $\mathbf{Y}$ or $\mathbf{Z}$
$\mathcal{K}_{\mathcal{U}}$	set of top-K relevant keywords for all the users' profiling results

4 FLOW-BASED CONSTRAINED CO-EMBEDDING MODEL

In this section, we introduce our flow-based constrained co-embeddings model, called FCCM, which aims to jointly learn the embeddings of both users and words in the same semantic space, such that the affinities between the two entities can be effectively measured for the user profiling task.

4.1 Preliminaries

The goal of our FCCM is to learn the semantic representations of two entities in QACs, namely the embeddings of users $\mathbf{Z} \in \mathbb{R}^{F\times D}$ with its u-th row $\mathbf{Z}_u$ being user u's embedding and word embeddings $\mathbf{Y} \in \mathbb{R}^{N\times D}$ with its w-th row $\mathbf{Y}_w$ being word w's embedding, respectively, where D, $N = |\mathcal{W}|$ and $F = |\mathcal{U}|$ denote the dimension of embeddings, the size of word vocabulary and the number of users, respectively.

To begin with our model description, we first briefly introduce the well-known VAE model [36], which is the basis of our proposed model. A VAE model is a probabilistic neural network model, which consists of two neural networks: 1) the inference network $q_\phi(z|x)$, namely the encoder, that infers the observed data into the latent distributional variables, 2) and the generative network $p_\theta(x|z)$, namely the decoder, that maps from latent distributional variables to the observed data. The training of a VAE model is to optimize the Evidence Lower BOund (denoted as ELBO) of the evidence likelihood $\log p_\theta(x)$:

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

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

However, there are three limitations when directly applying the vanilla VAE model to the user profiling task in QACs: (1) It can only learn embeddings of one type of entities, which is unable to capture the affinities between two categories of entities (e.g., in our setting we have users and words); (2) Standard VAEs usually assume that the variational posterior distribution is Gaussian, which may not always appropriate in many applications; (3) It is unable to integrate voting information into the model, which is an important indicator showing the quality of each answers as well as the expertise of the answerers.

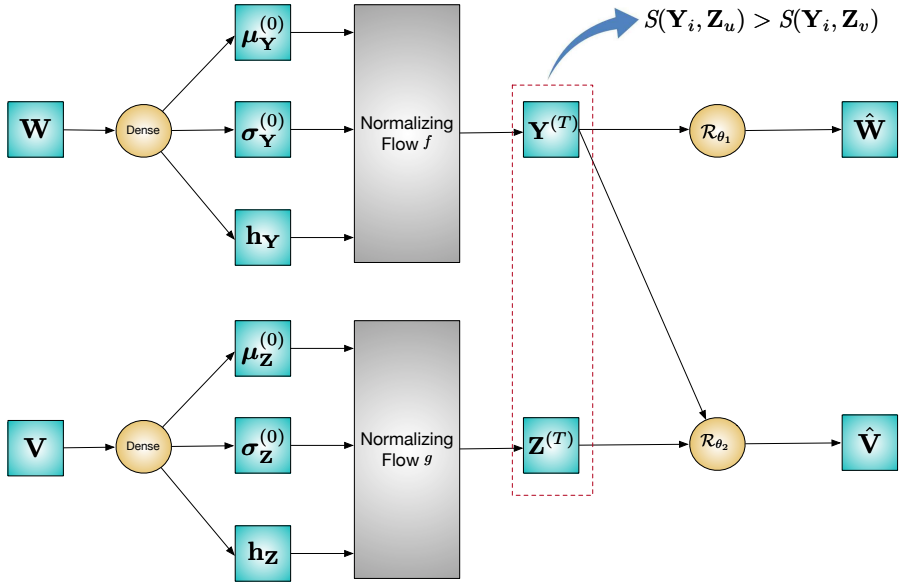


Fig. 1. The architecture of our FCCM. The inference model and the generative model act as the probabilistic encoder and the probabilistic decoder, respectively.

4.2 Overview of Our Model

Our FCCM model contains three main components: (1) Co-occurrence matrices and constraint information extraction; (2) Entity embedding training; (3) Users' expertise profiling; see Algorithm 1 for an overview. In "Part I" of Algorithm 1, we first extract the word-to-word and user-to-word co-occurrence matrices and voting constraints as the observation data of FCCM (steps 2-4 in Algorithm 1); In "Part II" of Algorithm 1, FCCM learns the embeddings of users and words by integrating VAEs and normalizing flow, such that the latent representations of users and words can be effectively mapped into the same semantic space for the user profiling task. Meanwhile, the SVM-Rank model is applied to make both user and word embeddings satisfy the constraints that constructed by the voting information (steps 6-21); In "Part III", we then use the learned embeddings of users and words to conduct an expertise-based ranking to find the top-K relevant keywords for each user (steps 23-25). The overall architecture of our FCCM is shown in Fig. 1. The model takes the word-to-word co-occurrence matrix $\mathbf{W}$ and the user-to-word co-occurrence matrix $\mathbf{V}$ as input and maps them to Gaussian distributions with means and variances as initial variational posteriors for all words and users, respectively. It then transforms the initial variational posteriors to the final latent embeddings by utilizing normalizing flows.

We next describe the "Part II" of Algorithm 1, which is the key component of FCCM. In particular, we detail how to co-embed users and words in §4.3, how to integrate normalizing flow in §4.4, and how to incorporate the voting constraints in §4.5, respectively.

4.3 Co-embedding Users and Words

To learn the embeddings of users and words, i.e., $\mathbf{Y}$ and $\mathbf{Z}$, we extract the word-to-word and user-to-word co-occurrence matrices, $\mathbf{W}$ and $\mathbf{V}$, from the answer sets generated by the users. In particular, to construct the word-to-word matrix $\mathbf{W}$, we first take the answers of all users as corpus, and then

ALGORITHM 1: Overview of our FCCM 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 $\mathcal{U}$

/* **Part I: Co-occurrence matrices and constraint information extraction** */
Construct matrix $\mathbf{W}$ by counting word-to-word co-occurrence information
Construct matrix $\mathbf{V}$ by counting user-to-word co-occurrence information
Construct constraints, $C_w = \{C_{w_1}, C_{w_2}, ..., C_{w_N}\}$, for each word using voting information $\mathcal{T}_{\mathcal{A}}$
/* **Part II: Entity embedding training by FCCM** */
$\theta^{(0)}$,$\phi^{(0)}$ $\leftarrow$ Initialize network parameters
$\boldsymbol{w}^{(0)}$,$b^{(0)}$ $\leftarrow$ Initialize SVM-Rank (similarity computing function $S(\cdot,\cdot)$) parameters as 0
repeat
  $\boldsymbol{\epsilon}_w \leftarrow \mathcal{N}(\mathbf{0}, \bar{\sigma}^2_{\mathcal{W}}\mathbf{I})$
  $\boldsymbol{\epsilon}_u \leftarrow \mathcal{N}(\mathbf{0}, \bar{\sigma}^2_{\mathcal{U}}\mathbf{I})$
  for $i = 1, \ldots, N$ **do**
    for $u, v \in C_{w_i}$ **do**
      $\mathbf{Y}^{(t)}, \mathbf{Z}^{(t)} \leftarrow \theta^{(t)}$
      construct constraints $C^i_{u,v} = S(\mathbf{Y}_i, \mathbf{Z}_u) - S(\mathbf{Y}_i, \mathbf{Z}_v)$
  $g_\theta, g_\phi \leftarrow \nabla_{\theta,\phi}\mathcal{L}(\theta, \phi; \mathbf{W}, \mathbf{V}, \boldsymbol{\epsilon}_w, \boldsymbol{\epsilon}_u) - \alpha \sum_{i=1}^{N} \sum_{(u,v)\in C_{w_i}} C^i_{u,v}$
  $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta g_\theta$
  $\phi^{(t+1)} \leftarrow \phi^{(t)} - \eta g_\phi$
  $\mathbf{Y}^{(t+1)}, \mathbf{Z}^{(t+1)} \leftarrow \theta^{(t+1)}$
  Construct training sample for SVM-Rank (e.g., $\{x :< \mathbf{Y}_i^{(t+1)}, \mathbf{Z}_u^{(t+1)} > - < \mathbf{Y}_i^{(t+1)}, \mathbf{Z}_v^{(t+1)} >, y : 1\}$)
  $\boldsymbol{w}^{(t+1)}, b^{(t+1)} \leftarrow$ optimize SVM-Rank to obtain parameters of similarity computing function
until *FCCM converges*
/* **Part III: Users' expertise profiling** */
for $u = 1, \ldots, F$ **do**
  $\mathcal{K}_u \leftarrow$ find top-K relevant words using embeddings produced by FCCM
Obtain the final user profiling results for all users $\mathcal{K}_{\mathcal{U}} = \{\mathcal{K}_{u_1}, \mathcal{K}_{u_2}, \ldots, \mathcal{K}_{u_F}\}$

count the number of co-occurrences between two words by setting a word context window with a fixed size. We construct the user-to-word matrix $\mathbf{V}$ by counting the number of co-occurrences between users and words such that $\mathbf{V}_{ij}$ records the number of times the i-th user and the j-th word co-occur in the same word context window. The co-occurrence information for user-to-word matrix can be defined in two ways, which will be discussed in §5.1.

The obtained word-to-word and user-to-word co-occurrence matrices will be feed as the input of our co-embedding model. The goal of the co-embedding model is to maximize the log-likelihood of these 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:

$$\log p(\mathbf{W}, \mathbf{V}) = \log \int_{\mathrm{Y}} \int_{\mathrm{Z}} p(\mathbf{W}, \mathbf{V}, \mathrm{Y}, \mathrm{Z}) \mathrm{dY} \mathrm{dZ}$$
$$= \log \int_{\mathrm{Y}} \int_{\mathrm{Z}} p(\mathbf{W}, \mathbf{V}, \mathrm{Y}, \mathrm{Z}) \frac{q_\phi(\mathrm{Y}, \mathrm{Z} \mid \mathbf{W}, \mathbf{V})}{q_\phi(\mathrm{Y}, \mathrm{Z} \mid \mathbf{W}, \mathbf{V})} \mathrm{dY} \mathrm{dZ}$$

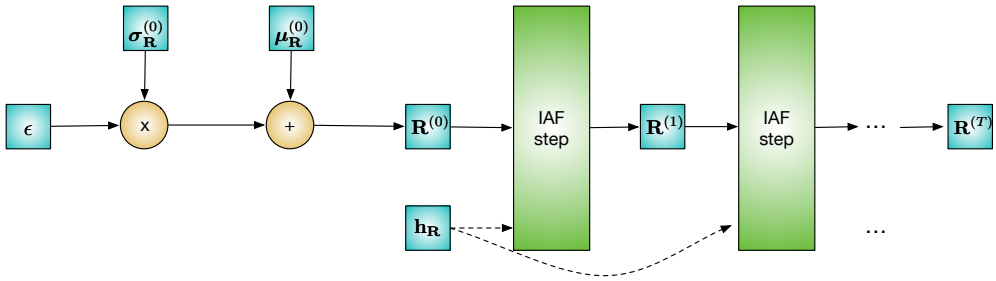


Fig. 2. The architecture of the IAF chain. It consists of a chain of invertible and smooth mappings with the initial variational posteriors $\mathbf{R}^{(0)}$ and context vector $\mathbf{h_R}$ as input, and obtain the final latent embeddings $\mathbf{R}^{(T)}$.

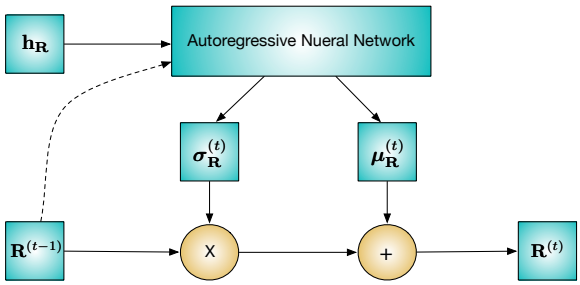


Fig. 3. Each IAF step in the mapping chain receives $\mathbf{R}^{(t-1)}$ and the context vector $\mathbf{h_R}$ as input, and produce $\mu_{\mathbf{R}}^{(t)}$ and $\sigma_{\mathbf{R}}^{(t)}$, then output $\mathbf{R}^{(t)}$.

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

where $p(\mathbf{W},\mathbf{V},\mathbf{Y},\mathbf{Z})$ is the joint distribution of latent variables and observations, with $\mathbf{Y}$ and $\mathbf{Z}$ being all words' latent embeddings and users' latent embeddings, respectively, and $q_\phi(\mathbf{Y},\mathbf{Z} \mid \mathbf{W},\mathbf{V})$ is the variational posterior over words and users for approximating the true posterior of latent variables $p(\mathbf{Y},\mathbf{Z} \mid \mathbf{W},\mathbf{V})$. Here ϕ is a set of trainable parameters in the inference model (i.e. encoder) to be estimated. For simplicity, $q_\phi(\mathbf{Y},\mathbf{Z} \mid \mathbf{W},\mathbf{V})$ is abbreviated as q_ϕ in the remainder of this paper.

The joint distribution $p(\mathbf{W},\mathbf{V},\mathbf{Y},\mathbf{Z})$ in Eq. (3) can be factorized as follows:

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

where $\theta = \{\theta_1,\theta_2\}$ with θ_1 and θ_2 representing the trainable parameters in the generative models (i.e. decoders) of words and users, respectively. $\mathbf{W}_{ij}$ is the co-occurrence information between word i and word j, with $\mathbf{Y}_i$ and $\mathbf{Y}_j$ 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 $\mathbf{Y}_i$ being the corresponding user embeddings and word embeddings, respectively.

For the variational posterior q_ϕ in Eq. (3) can be factorized in the following mean-field form:

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

where $\phi = \{\phi_1, \phi_2\}$ with ϕ_1 and ϕ_2 represent the trainable parameters of words and users in the inference model (i.e. encoders), respectively. Substituting Eq. (4) and Eq. (5) into Eq. (3) and introducing a trade-off parameter β, Eq. (3) can be represented as (the detailed derivation can be found in Appendix A):

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

where we have:

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

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

Here $\mathcal{L}(\theta, \phi; \mathbf{W}, \mathbf{V})$ is the ELBO on the marginal likelihood of the two co-occurrence matrices, which consists of the word-to-word loss $\mathcal{L}(\theta_1, \phi_1; \mathbf{W})$ and the user-to-word loss $\mathcal{L}(\theta_2, \phi_2; \mathbf{W}, \mathbf{V})$. In Eq. (7) and Eq. (8), the two inference models, i.e. $q_{\phi_1}(\mathbf{Y} \mid \mathbf{W})$ and $q_{\phi_2}(\mathbf{Z} \mid \mathbf{V})$, infer variational posterior distributions over the possible values of the latent embeddings $\mathbf{Y}$ and $\mathbf{Z}$, while the two generative models, i.e. $p_{\theta_1}(\mathbf{W}_{ij} \mid \mathbf{Y}_i, \mathbf{Y}_j)$ and $p_{\theta_2}(\mathbf{V}_{ui} \mid \mathbf{Z}_u, \mathbf{Y}_i)$, try to re-generate the possible values of observed users and words.

4.4 Co-embedding with Normalizing Flow

From section 4.3, we know that $\log p(\mathbf{W}, \mathbf{V})$ will reach its maximum when $D_{KL}(q_\phi \parallel p_\theta) = 0$, i.e., q_ϕ matches the true posterior distribution. However, it is not feasible to assume that the variational posterior distribution as Gaussian or other relatively simple distribution to approximate the true posterior because of the inherent limitations of the variational methods [62]. One solution to tackle this problem is to utilize normalizing flows [33, 35]. In this section, we will show how the normalizing flow framework is integrated into our FCCM model to infer better embeddings of users and words. In Figure 2, we provide a detailed architecture for the IAF chain. The constraints for $\mathbf{Z}_u, \mathbf{Z}_v$, and $\mathbf{Y}_i$ which are denoted as $S(\mathbf{Y}_i, \mathbf{Z}_u) > S(\mathbf{Y}_i, \mathbf{Z}_v)$ are adopted to pull users closer to the words they have expertise on in the semantic space.

There are two categories of entities in our task, i.e. users and words, and thus we employ two normalizing flows into the co-embedding models. Each flow is represented by a sequence of invertible and smooth mappings to describe the transformation of the probability density which can make the variational posteriors, i.e., $q_{\phi_1}(\mathbf{Y} \mid \mathbf{W})$ and $q_{\phi_2}(\mathbf{Z} \mid \mathbf{V})$, to be flexible enough to approximate the true posteriors. The two KL divergence terms in (6), i.e., $D_{KL}(q_{\phi_1}(\mathbf{Y} \mid \mathbf{W}) \parallel p(\mathbf{Y}))$ and $D_{KL}(q_{\phi_2}(\mathbf{Z} \mid \mathbf{V}) \parallel p(\mathbf{Z}))$, can be represented as follows:

$$\begin{aligned} KL &= D_{KL}(q_{\phi_1}(\mathbf{Y} \mid \mathbf{W}) \parallel p(\mathbf{Y})) + D_{KL}(q_{\phi_2}(\mathbf{Z} \mid \mathbf{V}) \parallel p(\mathbf{Z})) \\ &= \mathbb{E}_{q_\phi}\left[\log q_{\phi_1}(\mathbf{Y} \mid \mathbf{W}) - \log p_{\theta_1}(\mathbf{Y})\right] + \mathbb{E}_{q_{\phi_2}}\left[\log q_{\phi_2}(\mathbf{Z} \mid \mathbf{V}) - \log p_{\theta_2}(\mathbf{Z})\right] \end{aligned}$$

$$= \mathbb{E}_{q_\phi}\left[\sum_{i\in\mathcal{W}} \log q_{\phi_1}(\mathbf{Y}_i \mid \mathbf{W}) - \log p_{\theta_1}(\mathbf{Y}_i)\right] + \mathbb{E}_{q_\phi}\left[\sum_{u\in\mathcal{U}} \log q_{\phi_2}(\mathbf{Z}_u \mid \mathbf{V}) - \log p_{\theta_2}(\mathbf{Z}_u)\right]. \tag{9}$$

In order to make the derivation process convenient, we temporarily hide the trade-off parameter β in Eq. (9) and the remaining derivations in this section. For each word i and user u, we denote their initial variational posteriors as $q_{\phi_1}(\mathbf{Y}_i^{(0)} \mid \mathbf{W})$ and $q_{\phi_2}(\mathbf{Z}_u^{(0)} \mid \mathbf{V})$, respectively. Then, a sequence of invertible and smooth mappings are applied to transform each initial variational posterior to the final variational posterior:

$$\begin{aligned} \mathbf{Y}_i^{(t)} &= f_t\left(\mathbf{Y}_i^{(t-1)}\right) \quad t = 1, 2, \ldots, T, \\ \mathbf{Z}_u^{(t)} &= g_t\left(\mathbf{Z}_u^{(t-1)}\right) \quad t = 1, 2, \ldots, T, \end{aligned} \tag{10}$$

where $\mathbf{Y}_i^{(t)}$ and $\mathbf{Z}_u^{(t)}$ represent the t-th mapping results of the word i and user u, respectively, which are obtained by using the t-1-th mapping results as input by their corresponding mapping functions $f_t(\cdot)$ and $g_t(\cdot)$, respectively. The superscript (t) denotes that the initial variational posterior has been processed by t mappings, and T is the total number of mappings in a normalizing flow. Then, we can obtain the final variational posterior of words and users, i.e. $q_{\phi_1}(\mathbf{Y}_i^{(T)} \mid \mathbf{W})$ and $q_{\phi_2}(\mathbf{Z}_u^{(T)} \mid \mathbf{V})$, respectively. Thus, the overall KL divergence term of Eq. (9) can be rewritten as follows:

$$KL = \mathbb{E}_{q_\phi}\left[\sum_{i\in\mathcal{W}} \log q_{\phi_1}(\mathbf{Y}_i^{(T)} \mid \mathbf{W}) - \log p_{\theta_1}(\mathbf{Y}_i)\right] + \mathbb{E}_{q_\phi}\left[\sum_{u\in\mathcal{U}} \log q_{\phi_2}(\mathbf{Z}_u^{(T)} \mid \mathbf{V}) - \log p_{\theta_2}(\mathbf{Z}_u)\right]. \tag{11}$$

Substitute Eq. (11) into Eq. (6), the ELBO can be further represented as:

$$\begin{aligned} \mathcal{L}(\theta, \phi; \mathbf{W}, \mathbf{V}) = {} & \mathbb{E}_{q_\phi}\left[\sum_{(i,j)\in\mathcal{E}_\mathcal{W}} \log p_{\theta_1}(\mathbf{W}_{ij} \mid \mathbf{Y}_i, \mathbf{Y}_j)\right] \\ & + \mathbb{E}_{q_\phi}\left[\sum_{(u,i)\in\mathcal{E}_\mathcal{U}} \log p_{\theta_2}(\mathbf{V}_{ui} \mid \mathbf{Z}_u, \mathbf{Y}_i)\right] \\ & - \mathbb{E}_{q_\phi}\left[\sum_{i\in\mathcal{W}} \log q_{\phi_1}(\mathbf{Y}_i^{(T)} \mid \mathbf{W}) - \log p_{\theta_1}(\mathbf{Y}_i)\right] \\ & - \mathbb{E}_{q_\phi}\left[\sum_{u\in\mathcal{U}} \log q_{\phi_2}(\mathbf{Z}_u^{(T)} \mid \mathbf{V}) - \log p_{\theta_2}(\mathbf{Z}_u)\right]. \end{aligned} \tag{12}$$

According to the rule for transformation of densities [62], the variational posteriors in Eq. (12), i.e. $\log q_{\phi_1}(\mathbf{Y}_i^{(T)} \mid \mathbf{W})$ and $\log q_{\phi_2}(\mathbf{Z}_u^{(T)} \mid \mathbf{V})$, can be rewritten as follows:

$$\log q_{\phi_1}\left(\mathbf{Y}_i^{(T)}|\mathbf{W}\right) = \log q_{\phi_1}\left(\mathbf{Y}_i^{(0)}|\mathbf{W}\right) - \sum_{t=1}^{T} \log \left| \frac{d\mathbf{Y}_i^{(t)}}{d\mathbf{Y}_i^{(t-1)}} \right| \tag{13}$$

$$\log q_{\phi_2}\left(\mathbf{Z}_u^{(T)}|\mathbf{V}\right) = \log q_{\phi_2}\left(\mathbf{Z}_u^{(0)}|\mathbf{V}\right) - \sum_{t=1}^{T} \log \left| \frac{d\mathbf{Z}_u^{(t)}}{d\mathbf{Z}_u^{(t-1)}} \right|, \tag{14}$$

where $\frac{d\mathbf{Y}_i^{(t)}}{d\mathbf{Y}_i^{(t-1)}}$ and $\frac{d\mathbf{Z}_u^{(t)}}{d\mathbf{Z}_u^{(t-1)}}$ are both Jacobian matrices. The derivation of Eq. (13) and Eq. (14) are shown in Appendix B. The second RHS terms in Eq. (13) and Eq. (14) are the so-called "Log Determinant of Jacobia" terms, abbreviated as "LDJ". We should choose the mappings in the flow carefully, so that the LDJ terms can be easily computed.

In what follows, we detail how we further derive the final ELBO (see Eq. (22)) that can be represented as a close form expression by using the Inverse Autoregressive Flow (IAF) [35]. To facilitate further derivation, we start with the important definition in IAF:

Definition 1: If a vector $\mathbf{x} = (x_1, \ldots, x_D)$ is auto-regressively depend on another vector $\mathbf{y} = (y_1, \ldots, y_D)$, then the following relationship exists between them:

$$x_i = f_{x_i}\left(y_1, y_2, \cdots, y_{i-1}\right) \quad \forall i \in (1, D], \tag{15}$$

where f_{x_i} is a function mapping $(y_1, y_2, \cdots, y_{i-1})$ to x_i. We denote such relation between x and y as $\mathbf{x} \mapsto \mathbf{y}$.

According to *Definition 1*, we can derive directly that for any $j \geqslant i$, the partial derivative of x_i with respect to y_j is 0, i.e., $\frac{\partial x_i}{\partial y_j} = 0, j \geqslant i$. Thus the Jacobian matrix $\frac{dx}{dy}$ is a triangle matrix with all main-diagonal elements being 0, so that the determinant of $\frac{dx}{dy}$, i.e. $|\frac{dx}{dy}|$, will be equal to 0. The autoregressive character also has transitivity, which means if $\mathbf{x} \mapsto \mathbf{y}$ and $\mathbf{y} \mapsto \mathbf{z}$, then $\mathbf{x} \mapsto \mathbf{z}$.

We now go into the details of the derivation of the final ELBO. For convenience, we use the notation $\mathbf{R}$ to represent either $\mathbf{Y}$ or $\mathbf{Z}$ in the rest of paper. The IAF consists of a chain of T of the following transformations:

$$\mathbf{R}_i^{(t)} = \boldsymbol{\mu}_{\mathbf{R}_i}^{(t)} + \boldsymbol{\sigma}_{\mathbf{R}_i}^{(t)} \odot \mathbf{R}_i^{(t-1)} \quad t = 1, 2, \ldots, T, \tag{16}$$

where the mean $\boldsymbol{\mu}_{\mathbf{R}_i}^{(t)}$ and the variance $\boldsymbol{\sigma}_{\mathbf{R}_i}^{(t)}$ are the output of the autoregressive neural network at t-th step. We can know that both $\boldsymbol{\mu}_{\mathbf{R}_i}^{(t)}$ and $\boldsymbol{\sigma}_{\mathbf{R}_i}^{(t)}$ subject to *Definition 1* with $\mathbf{R}_i^{(t-1)}$, i.e. $\boldsymbol{\mu}_{\mathbf{R}_i}^{(t)} \mapsto \mathbf{R}_i^{(t-1)}$ and $\boldsymbol{\sigma}_{\mathbf{R}_i}^{(t)} \mapsto \mathbf{R}_i^{(t-1)}$. Thus, we can calculate the determinant of the Jacobian matrix of $\mathbf{R}_i^{(t)}$ w.r.t. $\mathbf{R}_i^{(t-1)}$ as:

$$\left| \frac{d\mathbf{R}_i^{(t)}}{d\mathbf{R}_i^{(t-1)}} \right| = \left| \frac{d\boldsymbol{\mu}_{\mathbf{R}_i}^{(t)}}{d\mathbf{R}_i^{(t-1)}} \right| + \left| \frac{d\boldsymbol{\sigma}_{\mathbf{R}_i}^{(t)}}{d\mathbf{R}_i^{(t-1)}} \right| \left| \mathbf{R}_i^{(t-1)} \right| + \left| \boldsymbol{\sigma}_{\mathbf{R}_i}^{(t)} \mathbf{I} \right|$$

$$= 0 + 0 \cdot \left|\mathbf{R}_i^{(t-1)}\right| + \prod_{d=1}^{D} \boldsymbol{\sigma}_{\mathbf{R}_i}^{(t)} \Bigg|_d = \prod_{d=1}^{D} \boldsymbol{\sigma}_{\mathbf{R}_i}^{(t)} \Bigg|_d, \tag{17}$$

where $\bullet\,|_d$ denotes the d-th element of $\bullet$. Fig. 3 visualizes how we obtain $\mathbf{R}^{(t)}$ from the previous embedding $\mathbf{R}^{(t-1)}$ and its context embedding $\mathbf{h}_{\mathbf{R}}$ at the flow. Then we assume the initial variational posterior of each word and user are simply diagonal Gaussian distributions:

$$q_{\phi_1}(\mathbf{Y}_i^{(0)} \mid \mathbf{W}) = \mathcal{N}(\mathbf{Y}_i^{(0)} | \boldsymbol{\mu}_{\mathbf{Y}_i}, \boldsymbol{\sigma}_{\mathbf{Y}_i}^2 \mathbf{I}), \tag{18}$$

$$q_{\phi_2}(\mathbf{Z}_u^{(0)} \mid \mathbf{V}) = \mathcal{N}(\mathbf{Z}_u^{(0)} | \boldsymbol{\mu}_{\mathbf{Z}_u}, \boldsymbol{\sigma}_{\mathbf{Z}_u}^2 \mathbf{I}), \tag{19}$$

where $\boldsymbol{\mu}_{\mathbf{Y}_i}$ and $\boldsymbol{\mu}_{\mathbf{Z}_u}$ $\boldsymbol{\sigma}_{\mathbf{Y}_i}^2$ and $\boldsymbol{\sigma}_{\mathbf{Z}_u}^2$ are means and variances of the initial variational posterior of words and users to be learned, respectively. The prior of the final embedding can be represented as a standard diagonal Gaussian distribution:

$$p(\mathbf{Y}_i^{(T)}) = \mathcal{N}(\mathbf{Y}_i^{(T)} | \mathbf{0}, \mathbf{I}), \tag{20}$$

$$p(\mathbf{Z}_u^{(T)}) = \mathcal{N}(\mathbf{Z}_u^{(T)} | \mathbf{0}, \mathbf{I}), \tag{21}$$

Then we obtain the form of ELBO with normalizing flow from $t = 0$ to $t = T$ (the detailed derivation is provided in Appendix C) as:

$$\begin{aligned}
\mathcal{L}(\theta, \phi; \mathbf{W}, \mathbf{A}) = \mathbb{E}_{q_\phi} & \left[\sum_{(i,j) \in \mathcal{E}_{\mathcal{W}}} \log p_\theta \left(\mathbf{W}_{ij} | \mathbf{Y}_i^{(T)}, \mathbf{Y}_j^{(T)} \right) \right] \\
+ \beta \mathbb{E}_{q_\phi} & \left[\sum_{(u,i) \in \mathcal{E}_{\mathcal{U}}} \log p_\theta \left(\mathbf{V}_{ui} | \mathbf{Z}_u^{(T)}, \mathbf{Y}_i^{(T)} \right) \right] \\
- \mathbb{E}_{q_\phi} & \left[\sum_{i \in \mathcal{W}} \sum_{d=1}^{D} \left(-\frac{1}{2} \boldsymbol{\epsilon}_{\mathbf{Y}_i}^2 - \sum_{t=0}^{T} \log \boldsymbol{\sigma}_{\mathbf{Y}_i}^{(t)} + \frac{1}{2} \left(\mathbf{Y}_i^{(T)} \right)^2 \right) \Bigg|_d \right] \\
- \beta \mathbb{E}_{q_\phi} & \left[\sum_{u \in \mathcal{U}} \sum_{d=1}^{D} \left(-\frac{1}{2} \boldsymbol{\epsilon}_{\mathbf{Z}_u}^2 - \sum_{t=0}^{T} \log \boldsymbol{\sigma}_{\mathbf{Z}_u}^{(t)} + \frac{1}{2} \left(\mathbf{Z}_u^{(T)} \right)^2 \right) \Bigg|_d \right],
\end{aligned} \tag{22}$$

where $\boldsymbol{\epsilon}_{\mathbf{Y}_i} = \frac{\mathbf{Y}_i^{(0)} - \boldsymbol{\mu}_{\mathbf{Y}_i}^{(0)}}{\boldsymbol{\sigma}_{\mathbf{Y}_i}^{(0)}}$ and $\boldsymbol{\epsilon}_{\mathbf{Z}_u} = \frac{\mathbf{Z}_u^{(0)} - \boldsymbol{\mu}_{\mathbf{Z}_u}^{(0)}}{\boldsymbol{\sigma}_{\mathbf{Z}_u}^{(0)}}$. β is the trade-off parameter we mentioned in (6). Finally, we can simply derivate from Monte Carlo estimates of these expectation terms by the following estimators for our model as:

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

$$
\begin{aligned}
&- \frac{1}{L} \sum_{i \in \mathcal{W}} \sum_{d=1}^{D} \sum_{l=1}^{L} \left(-\frac{1}{2} \epsilon_{\mathbf{Y}_i}^2 - \sum_{t=0}^{T} \log \sigma_{\mathbf{Y}_i}^{(t,l)} + \frac{1}{2} \left(\mathbf{Y}_i^{(T,l)} \right)^2 \right) \Bigg|_d \\
&- \frac{\beta}{L} \sum_{u \in \mathcal{U}} \sum_{d=1}^{D} \sum_{l=1}^{L} \left(-\frac{1}{2} \epsilon_{\mathbf{Z}_u}^2 - \sum_{t=0}^{T} \log \sigma_{\mathbf{Z}_u}^{(t,l)} + \frac{1}{2} \left(\mathbf{Z}_u^{(T,l)} \right)^2 \right) \Bigg|_d ,
\end{aligned} \tag{23}
$$

where L is the sampling size.

4.5 Modelling Constraints of Voting

In order to enforce the embeddings of users and words subject to the voting constraint during the inference of the embeddings of users and word, we formally define a single constraint that that user u has more votes than user v for word i as:

$$
S(\mathbf{Y}_i, \mathbf{Z}_u) \geq S(\mathbf{Y}_i, \mathbf{Z}_v), \tag{24}
$$

where $\mathbf{Y}_i$, $\mathbf{Z}_u$ and $\mathbf{Z}_v$ represent the embeddings of word i, user u and user v, respectively. The similarity function $S(\cdot, \cdot)$ is a scoring function that is introduced to compute the similarity between user embeddings and word embeddings.

To incorporate such constraints into our user profiling task as we describe above, we extends the co-embedding model by using Max-margin methods. The max-margin methods, such as SVM-Rank [24], have been widely applied into many information retrieval methods. In particular, given a set of training queries $\{q_i\}_{i=1}^n$, with n being the number of queries, their associated document pairs $(\boldsymbol{x}_u^{(i)}, \boldsymbol{x}_v^{(i)}) \in \mathcal{S}_i$ and the corresponding ground-truth label $y_{u,v}^{(i)}$, the standard SVM-Rank model try to optimize the following objective function:

$$
\begin{aligned}
&\min_{\boldsymbol{w}, b, \xi} \frac{1}{2} ||\boldsymbol{w}||_2^2 + \lambda \sum_{i=1}^{n} \sum_{(u,v) \in \mathcal{S}_i} \xi_{(u,v)}^i \\
&s.t. \quad y_{u,v}^{(i)} \left[\boldsymbol{w}^T (\boldsymbol{x}_u^{(i)} - \boldsymbol{x}_v^{(i)}) + b \right] \geq 1 - \xi_{(u,v)}^i, \\
&\qquad \xi_{(u,v)}^i \geq 0, \quad i = 1, ..., n,
\end{aligned} \tag{25}
$$

where $\xi_{(u,v)}^i$ is a slack variable representing the tolerance of incorrect ranking document u and document v on query i. In the document ranking task, a scoring function is used to compute the relevance between query i and document u, i.e., $f(q_i, \boldsymbol{x}_u^i) = \boldsymbol{w}^T \boldsymbol{x}_u^{(i)} + b$.

The SVM-Rank [24] constraint formulation can also be applied in our task. Specifically, given all N words, i.e., the size of vocabulary in our task, their associated 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 its corresponding label $y_{u,v}^{(i)}$, we can construct the following optimization function:

$$
\begin{aligned}
&\min_{\boldsymbol{w}, b, \xi} \mathcal{L}_{SVM} = \min_{\boldsymbol{w}, b, \xi} \frac{1}{2} ||\boldsymbol{w}||_2^2 + \lambda \sum_{i=1}^{N} \sum_{(u,v) \in \mathcal{S}_{w_i}} \xi_{(u,v)}^i \\
&s.t. \quad y_{u,v}^{(i)} \left[\boldsymbol{w}^T (< \mathbf{Y}_i, \mathbf{Z}_u > - < \mathbf{Y}_i, \mathbf{Z}_v >) + b \right] \geq 1 - \xi_{(u,v)}^i \\
&\qquad \xi_{(u,v)}^i \geq 0, \quad i = 1, ..., N,
\end{aligned} \tag{26}
$$

where $\xi_{(u,v)}^i$ 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. Unlike the standard SVM-Rank, the scoring function in (26) receives two inputs, i.e., user embeddings and word embeddings, and if one user has high expertise for a specific word, their embeddings should

have high similarity, such that the scoring function should output a high score number when receiving their embeddings as input. Therefore the scoring function can be seen as the function of computing the similarity between user embeddings and word embeddings. Specifically, 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: $S(\mathbf{Y}_i, \mathbf{Z}_u) = \boldsymbol{w}^T < \mathbf{Y}_i, \mathbf{Z}_u > +b$. Then we can obtain the parameters of the scoring function $S(\cdot, \cdot)$, i.e., $\boldsymbol{w}$ and b, by solving the quadratic programming problem as discussed above.

To obtain the embeddings of both users and words with additional constraints, FCCM optimizes a combined loss of ELBO and the max-margin classifier of SVM-Rank. The final objective of the learning for our FCCM can be written as follows:

$$\begin{aligned} &\min_{\theta,\phi,\boldsymbol{w},b,\xi} \mathcal{L} = \min_{\theta,\phi,\boldsymbol{w},b,\xi} \mathcal{L}(\theta,\phi;\mathbf{W},\mathbf{V}) + \mathcal{L}_{SVM} \\ &s.t. \quad y_{a,b}^{(i)} \left[\boldsymbol{w}^T (< \mathbf{Y}_i, \mathbf{Z}_u > - < \mathbf{Y}_i, \mathbf{Z}_v >) + b\right] \geq 1 - \xi_{(u,v)}^i \\ &\quad \xi_{(u,v)}^i \geq 0, \quad i = 1, ..., n. \end{aligned} \tag{27}$$

5 OPTIMIZATION

To optimize Eq. (27), we need update two parts of the trainable parameters, namely, the parameters in the VAE loss $\mathcal{L}(\theta, \phi; \mathbf{W}, \mathbf{V})$ (the encoders ϕ and decoders θ) and the parameters in the SVM-Rank loss $\mathcal{L}_{SVM}$ $(\boldsymbol{w}, b, \xi)$. In order to jointly train our model, we first fix the parameters of the VAE loss (i.e. ϕ and θ), 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 VAE loss. We iteratively update the parameters of these two parts until coverage, such that the variational latent variables subject to the constraints while still being able to co-embed users and words in the same semantic space.

5.1 Optimize θ and ϕ

To update the parameters of the VAE loss, we fix $\boldsymbol{w}$ and b and optimize the following objective function:

$$\begin{aligned} &\min_{\theta,\phi} \mathcal{L}(\theta,\phi;\mathbf{W},\mathbf{V}) - \alpha \left[\sum_{i=1}^{N} \sum_{(u,v)\in C_{w_i}} S(\mathbf{Y}_i, \mathbf{Z}_u) - S(\mathbf{Y}_i, \mathbf{Z}_v)\right] \\ &s.t. \quad \alpha \geq 0, \end{aligned} \tag{28}$$

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. $\mathbf{Y}_i$) 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. (28) 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. (28), we apply two neural network models, i.e., the *inference model* with trainable parameters ϕ and the *generative model* with trainable parameters θ, to perform gradient decent for learning all the model parameters.

Inference Model. In this part, we aim at encoding from observation variables, i.e., word-to-word co-occurrence matrix $\mathbf{W}$ and user-to-word co-occurrence matrix $\mathbf{V}$, to initial variational posteriors. We apply two two-layer fully connected neural networks as our two encoders, one for inferring the initial variational posteriors of the users and another for inferring the initial variational posteriors

of the words, respectively. Specifically, the network structure is defined as follows:

$$\begin{aligned}\mathbf{H}_{\mathcal{U}}^{(1)} &= \text{relu}\left(\mathbf{V}\mathbf{W}_{\mathcal{U}}^{(o)} + \mathbf{b}_{\mathcal{U}}^{(o)}\right),\\ \left[\boldsymbol{\mu}_{\mathbf{Z}}^{(0)}, \boldsymbol{\sigma}_{\mathbf{Z}}^{(0)}, \mathbf{h}_{\mathbf{Z}}\right] &= \mathbf{H}_{\mathcal{U}}^{(1)}\mathbf{W}_{\mathcal{U}}^{(1)} + \mathbf{b}_{\mathcal{U}}^{(1)},\end{aligned} \tag{29}$$

$$\begin{aligned}\mathbf{H}_{\mathcal{W}}^{(1)} &= \text{relu}\left(\mathbf{W}\mathbf{W}_{\mathcal{W}}^{(o)} + \mathbf{b}_{\mathcal{W}}^{(o)}\right),\\ \left[\boldsymbol{\mu}_{\mathbf{Y}}^{(0)}, \boldsymbol{\sigma}_{\mathbf{Y}}^{(0)}, \mathbf{h}_{\mathbf{Y}}\right] &= \mathbf{H}_{\mathcal{W}}^{(1)}\mathbf{W}_{\mathcal{W}}^{(1)} + \mathbf{b}_{\mathcal{W}}^{(1)},\end{aligned} \tag{30}$$

where $\boldsymbol{\mu}_{\mathbf{Y}}^{(0)}$ and $\boldsymbol{\sigma}_{\mathbf{Y}}^{(0)}$ are the means and variances of the initial variational posteriors of words, $\boldsymbol{\mu}_{\mathbf{Z}}^{(0)}$ and $\boldsymbol{\sigma}_{\mathbf{Z}}^{(0)}$ are the means and variances of the initial variational posteriors of users, and $\mathbf{h}_{\mathbf{Y}}$ and $\mathbf{h}_{\mathbf{Z}}$ are the context vector of words and users, respectively. $\mathbf{b}$ is the bias and $\text{relu}(\cdot)$ is the rectified linear unit activation function [21]. For brevity, we let $\phi = [\phi_1, \phi_2]$ denote all the trainable parameters of encoder with $\phi_1 = [\mathbf{W}_{\mathcal{W}}^{(o)}, \mathbf{W}_{\mathcal{W}}^{(1)}, \mathbf{b}_{\mathcal{W}}^{(o)}, \mathbf{b}_{\mathcal{W}}^{(1)}]$ being the trainable weights for the word inference layers and $\phi_2 = [\mathbf{W}_{\mathcal{U}}^{(o)}, \mathbf{W}_{\mathcal{U}}^{(1)}, \mathbf{b}_{\mathcal{U}}^{(o)}, \mathbf{b}_{\mathcal{U}}^{(1)}]$ being trainable weights for the user inference layers.

After having obtained all the means and variances for all the initial variational posteriors of words and users, the reparameterization trick [36] is applied to transform the latent Gaussian random variables to the deterministic values of $\mathbf{Y}^{(0)}$ and $\mathbf{Z}^{(0)}$, i.e., initial word embeddings and initial user embeddings.

Inverse Autoregressive Flows. The part of IAF aims to transform the initial variational posteriors to approximate the true posteriors. The masked matrix [20] is used to implement the "autoregressive" feature. Each invertible and smooth mapping in IAF chain is defined as follows:

$$\mathbf{H}_{\mathcal{I}}^{(1)} = \text{ELU}\left(\mathbf{R}^{(t-1)}\left(\mathbf{M} \odot \mathbf{W}_{\mathcal{I}}^{(0)}\right) + \mathbf{b}_{\mathcal{I}}^{(0)}\right), \tag{31}$$

$$\mathbf{H}_{\mathcal{I}}^{(1)} = \mathbf{H}_{\mathcal{I}}^{(1)} + \mathbf{h}_{\mathbf{R}}, \tag{32}$$

$$\mathbf{H}_{\mathcal{I}}^{(2)} = \text{ELU}\left(\mathbf{H}_{\mathcal{I}}^{(1)}\left(\mathbf{M} \odot \mathbf{W}_{\mathcal{I}}^{(1)}\right) + \mathbf{b}_{\mathcal{I}}^{(1)}\right), \tag{33}$$

$$\boldsymbol{\mu}_{\mathbf{R}}^{(t)} = \mathbf{H}_{\mathcal{I}}^{(2)}\left(\mathbf{M}' \odot \mathbf{W}_{\mathcal{I}}^{(2)}\right) + \mathbf{b}_{\mathcal{I}}^{(2)}, \tag{34}$$

$$\boldsymbol{\sigma}_{\mathbf{R}}^{(t)} = \text{sigmoid}\left[\mathbf{H}_{\mathcal{I}}^{(2)}\left(\mathbf{M}' \odot \mathbf{W}_{\mathcal{I}}^{(3)}\right) + \mathbf{b}_{\mathcal{I}}^{(3)} + s\right], \tag{35}$$

$$\mathbf{R}^{(t)} = \boldsymbol{\mu}_{\mathbf{R}}^{(t)} + \boldsymbol{\sigma}_{\mathbf{R}}^{(t)} \odot \mathbf{R}^{(t-1)}, \tag{36}$$

$$l_{\mathbf{R}} = l_{\mathbf{R}} + \log \boldsymbol{\sigma}_{\mathbf{R}}^{(t)}\mathbf{1}, \tag{37}$$

where $\text{ELU}(\cdot)$ is a non-linear activation function [3]:

$\mathbf{M}$ and $\mathbf{M}'$ are autoregressive mask matrix defined as follow:

If $\mathbf{M}, \mathbf{M}' \in \mathbb{R}^{d_1 \times d_2}$, $d_1 \geqslant d_2$ and $\frac{d_1}{d_2} = k, k \in \mathbb{Z}^+$, then

$$\mathbf{M}_{ij} = \begin{cases} 0 & i > k \cdot j, \\ 1 & i \leqslant k \cdot j, \end{cases} \tag{38}$$

$$\mathbf{M}'_{ij} = \begin{cases} 0 & i > k(j-1), \\ 1 & i \leqslant k(j-1). \end{cases} \tag{39}$$

If $d_2 \geqslant d_1$ and $\frac{d_2}{d_1} = k, k \in \mathbb{Z}^+$, then

$$\mathbf{M}_{ij} = \begin{cases} 0 & j <= k(i-1), \\ 1 & j > k(i-1), \end{cases} \tag{40}$$

$$\mathbf{M}'_{ij} = \begin{cases} 0 & j <= k \cdot i, \\ 1 & j > k \cdot i, \end{cases} \tag{41}$$

s is similar to the forget bias in Long Short-Term Memory (LSTM) that usually initialized with +1 or +2, and sigmoid$(\cdot)$ is represented as the standard sigmoid activation function. For calculate the KL divergence term, $l_{\mathbf{R}}$ is a global variable used to add up all LDJ terms. $\mathbf{1}$ is a vector of ones and in reality $\log \boldsymbol{\sigma}_{\mathbf{R}}^{(t)}$ is equal to add up all elements in each row. $\phi_{\mathbf{R}}^{(t)} = \left[\mathbf{W}_{\mathcal{I}}^{(0)}, \mathbf{W}_{\mathcal{I}}^{(1)}, \mathbf{W}_{\mathcal{I}}^{(2)}, \mathbf{W}_{\mathcal{I}}^{(3)}, \mathbf{b}_{\mathcal{I}}^{(0)}, \mathbf{b}_{\mathcal{I}}^{(1)}, \mathbf{b}_{\mathcal{I}}^{(2)}, \mathbf{b}_{\mathcal{I}}^{(3)}\right]$ are all trainable weights of the t-th autoregressive neural network in flow.

According to (31) and (33), we can know that $\mathbf{H}_{\mathcal{I}}^{(1)} \mapsto \mathbf{R}^{(t-1)}$ and $\mathbf{H}_{\mathcal{I}}^{(2)} \mapsto \mathbf{H}_{\mathcal{I}}^{(1)}$. According to (34) and (35), we can know that $\boldsymbol{\mu}_{\mathbf{R}}^{(t)} \mapsto \mathbf{H}_{\mathcal{I}}^{(2)}$ and $\boldsymbol{\sigma}_{\mathbf{R}}^{(t)} \mapsto \mathbf{H}_{\mathcal{I}}^{(2)}$. Then according to the transitivity of autoregressive character, we have:

$$\boldsymbol{\mu}_{\mathbf{R}}^{(t)} \mapsto \mathbf{R}^{(t-1)}, \tag{42}$$

$$\boldsymbol{\sigma}_{\mathbf{R}}^{(t)} \mapsto \mathbf{R}^{(t-1)}. \tag{43}$$

According to [35], for numerical stability, $\mathbf{R}^{(t)}$ is obtained by:

$$\mathbf{R}^{(t)} = \left(1 - \boldsymbol{\sigma}_{\mathbf{R}}^{(t)}\right) \odot \boldsymbol{\mu}_{\mathbf{R}}^{(t)} + \boldsymbol{\sigma}_{\mathbf{R}}^{(t)} \odot \mathbf{R}^{(t-1)}. \tag{44}$$

Note that it is a special form of updating the t-th mapping result of IAF chain, i.e. $\mathbf{R}^{(t)}$, and thus the simple computation of the log-determinant of Jacobian matrices still applies.

Generative Model. The generative model aims to reconstruct the observation variables, i.e., the word-to-word co-occurrence matrix $\mathbf{W}$ and the user-to-word co-occurrence matrix $\mathbf{V}$, according to their latent representations $\mathbf{Y}$ and $\mathbf{Z}$. Here we denote the word-to-word generative model network by $\mathcal{R}_{\theta_1}$. Then $\mathcal{R}_{\theta_1}$ takes latent embeddings of words $\mathbf{Y}_i$ and $\mathbf{Y}_j$ as input, and outputs the parameters of the corresponding distribution:

$$[\boldsymbol{\mu}_{\mathcal{E}_W(i,j)}, \boldsymbol{\sigma}^2_{\mathcal{E}_W(i,j)}] = \mathcal{R}_{\theta_1}(\mathbf{Y}_i, \mathbf{Y}_j). \tag{45}$$

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

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

$$p_{\theta_1}(\mathbf{W}_{ij} \mid \mathbf{Y}_i, \mathbf{Y}_j) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{E}_W(i,j)}, \boldsymbol{\sigma}^2_{\mathcal{E}_W(i,j)}\mathbf{I}). \tag{46}$$

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

$$p_{\theta_1}(\mathbf{W}_{ij} \mid \mathbf{Y}_i, \mathbf{Y}_j) = \mathrm{Ber}(\boldsymbol{\mu}_{\mathcal{E}_W(i,j)}). \tag{47}$$

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

Similarly, we denote the user-to-word generative model network by $\mathcal{R}_{\theta_2}$, which takes user embeddings $\mathbf{Z}_u$ and word embeddings $\mathbf{Y}_i$ as input, and outputs the parameters of the corresponding distribution:

$$[\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{U}}(u,i)}, \boldsymbol{\sigma}^2_{\mathcal{E}_{\mathcal{U}}(u,i)}] = \mathcal{R}_{\theta_2}(\mathbf{Z}_u, \mathbf{Y}_i). \tag{48}$$

$\mathbf{V}_{ui}$ 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 information, e.g., the probability that word i is associated with user u can be:

$$p_{\theta_1}(\mathbf{V}_{ui} \mid \mathbf{Z}_u, \mathbf{Y}_i) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{U}}(u,i)}, \boldsymbol{\sigma}^2_{\mathcal{E}_{\mathcal{U}}(u,i)}\mathbf{I}). \tag{49}$$

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

$$p_{\theta_1}(\mathbf{V}_{ui} \mid \mathbf{Z}_u, \mathbf{Y}_i) = \mathrm{Ber}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{U}}(u,i)}). \tag{50}$$

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

$$\begin{aligned} \mathcal{R}_{\theta_1}(\mathbf{Y}_i, \mathbf{Y}_j) &= \mathrm{sigmoid}(\mathbf{Y}_i^{\mathrm{T}}\mathbf{Y}_j), \\ \mathcal{R}_{\theta_2}(\mathbf{Z}_u, \mathbf{Y}_i) &= \mathrm{sigmoid}(\mathbf{Z}_u^{\mathrm{T}}\mathbf{Y}_i), \end{aligned} \tag{51}$$

where $\mathrm{sigmoid}(\cdot)$ is the sigmoid function.

5.2 Optimize w and b

To update the parameters of the SVM-Rank objective, we fix θ and ϕ, then the optimization of objective Eq. (27) is equivalent to optimizing the SVM-Rank objective (see §4.5). In particular, we can update the weight vector $\boldsymbol{w}$ and the bias b, i.e., the parameters of the similarity computing function $S(\cdot,\cdot)$, by solving the quadratic programming problem.

After obtaining the embeddings of all users and words, we can calculate the cosine similarities between all the user-word pairs, and retrieve the top-K relevant keywords, i.e., $\mathcal{K}_u$, based on the similarity scores.

6 EXPERIMENTAL SETUP

In this section, we detail our experimental setup. In Section § 6.1, we fist list the research questions that guide the remainder of the paper. Then, we describe the datasets, baselines and evaluation metrics in Section § 6.2, § 6.3 and § 6.4, respectively.

6.1 Research Questions

We target at answering the below research questions that guide the remainder of the paper:

(**RQ1**) Can our proposed FCCM model outperform the state-of-the-art baseline models in terms of the user profiling task in QACs?

(**RQ2**) Can the integration of the additional constraints enhance the performance of our FCCM for the user profiling task in QACs?

(**RQ3**) Can the vector representations of users and words inferred by our FCCM model effectively measure the semantic affinities between them?

(**RQ4**) What is the impact of the normalizing flow integrated into our FCCM model on the performance of our FCCM model for the user profiling task in QACs?

(**RQ5**) How good is the quality of the semantic embeddings obtained by our FCCM model in terms of generalization performance?
(**RQ6**) What is the impact of the trade-off hyper-parameters on the quality of inferred representations of users and words?
(**RQ7**) How do the embedding dimensions of users and words impact the performance of our FCCM model in the user profiling task?

6.2 Dataset

Due to the lack of related research on user profiling tasks in QACs, there is no relevant public available dataset that is suitable to train and test our proposed model. To evaluate the effectiveness of our proposed model, we extract a real-world QAC dataset in the experiment, by crawling a large number of relevant question answering data from a well-known Chinese question answering community platform, Zhihu, where questions are posted and answered by its community users [3]. Specifically, the overall extraction process is as follows:

(1) **Seed users selection**. We manually selected a number of users who have expertise on one or multiple fields, such as sport, music and astronomy etc. Specifically, we selected users according to the following criterion: (1) the number of users' answers must be ≥ 100; (2) the total number of votes received by users must be $\geq 10{,}000$; (3) the number of user's followers must be ≥ 1000. After filtering out the users according to the above conditions, we then crawled all answers they posed and the votes of these answers.
(2) **Data cleaning**. We mainly remove meaningless content in the answers, such as URL links and image links, as well as some informal answers (some answers are just a joke or other content that does not reflect the corresponding user's expertise).
(3) **Word segmentation and filtering**. Specifically, we first segmented answers posted by all selected users into words by the using Chinese word segment tool *pkuseg* [67]. 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 2,692 users, about 30,000 words, and the corresponding votes associated with each word.
(4) **Annotation**. To do this, we employ a number of annotators. For each user, each 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, 98 annotators took part in the annotation, with each of them labelling about 25 users.

Through the above process, we can finally obtain the ground truth profile of all selected users. Table 3 shows the ground truth profile of three users with user id 13, 22, and 64, respectively. Note that here for brevity, we only list the Top-10 keywords for each user. From the keywords corresponding to each user in Table 3, it can be seen that the main areas profiled by user 13 are the Internet and manufacturing, user 22 is more interested in politics and international situations, while user 64 is interested in finance, real estate, and basketball.

Since there is a heavy workload on the data annotation, in this paper we only focus on the Zhihu dataset, which is a Chinese dataset, and we leave the evaluation of datasets over other languages and other QAC platforms as future works. In order to validate the generalization of our proposed models, we further split the users into four subsets over different topics. To obtain such subsets, we use the Author Topic Model (ATM) [64] to cluster the users into five groups, inspired by [75]. The split subsets and their top-k highest probability words per topic are shown in Table 2.

[3]The dataset used in our experiments is publicly downloadable from https://bitbucket.org/luoyp21/zhihuq-a/get/master.zip.

Table 2. Top words of the five subsets. Note that some stop words are manually removed from the list.

Group/Topic	#1	#2	#3	#4	#5
# Users	532	456	554	378	772
Top-5 Words	英国(UK)	训练(Training)	智能(Intelligence)	爱情(Love)	皮肤(Skin)
	投资(Invest)	比赛(Competition)	大数据(Big data)	焦虑(Anxiety)	公众号(Account)
	资本(Capital)	热量(Calories)	互联网(Internet)	治疗(Therapy)	品牌(Brand)
	银行(Bank)	食物(Food)	运营(Operation)	社交(Soical)	开发(Development)
	金融(Finance)	体系(System)	品牌(Brand)	婚姻(Marriage)	文明(Culture)

Table 3. The ground truth profiles for three users with user ID #13, #22, and #64, respectively. Words are translated into English in the corresponding brackets.

User ID	#13	#22	#61
Ground truth	互联网 (WWW)	国家 (Countries)	楼市 (Estate market)
	制造业 (Industries)	中国 (China)	房地产 (Estate)
	阿里巴巴 (Alibaba)	中国人 (Chinese)	篮球 (Basketball)
	腾讯 (Tecent)	外国 (Foreign countries)	婚姻 (Marriage)
	亚马逊 (Amazon)	美国 (US.)	股票 (Stocks)
	宜家 (IKEA)	新闻 (News)	姚明 (Yao Ming)
	零售 (Retail)	法律 (Laws)	科比 (Kobe Bean Bryant)
	富士康 (Foxconn)	世界 (The world)	学区房 (School district houses)
	马云 (Jack Ma)	主权 (Sovereignty)	基金 (Funds)
	广州 (Guangzhou city)	印度 (India)	珠海 (Zhuhai city)

6.3 Baselines and Settings

We compare our proposed FCCM model with the following 10 baseline algorithms for the task of user profiling:

(i) **TF-IDF:** This baseline model applies the traditional TF-IDF algorithm [73] to obtain the retrieval scores of words given a user's associated documents (in our setting, the answers) and regards those top-N retrieval words as the results for profiling the user.

(ii) **Average Word Embedding (AWE):** This baseline model takes the average of the word embeddings in the answers associated with the user as the semantic representation of the user. The word embeddings here are obtained from the Chinese word embeddings pre-trained by Tencent's Directional Skip-Gram model [66].

(iii) **TF-IDF + Word Embedding (TWE):** This baseline model takes the TF-IDF scores as the weights of the words for a user, and then regards the average of the representations of the words that multiple the weights as the final semantic representations of the words for the users. The users are then profiled by the top-N words with the top-N retrieval scores. The word embeddings here are also obtained from the Chinese word embeddings pre-trained by Tencent's Directional Skip-Gram model [66].

(iv) **Word2Vec:** This is the traditional word embedding algorithm [54]. It regards each user as a special word and the words in the answers posted by the user as the context words.

(v) **Neural Variational Document Model (NVDM):** This is a neural document embedding model [52] that infers both the embeddings of documents and words. We utilize this model to obtain the representations of the answers, which are then averaged to be the representation of the user who posted these answers. The users are then profiled by the top-N most relative words that are most similar to the embeddings of the users.

(vi) **Embedding Topic Model (ETM):** This is a topic model that takes into account the embeddings of words [14]. The model co-embeds semantic representations of latent topics and words. The topic distributions of the answers associated with the users are regarded

as the semantic representations of the users. Words with their embeddings matching the embeddings of the users are taken as the profiling results of the users.

(vii) **Labelled ETM:** This is a variant of the ETM. "Labeled" denotes the word embeddings using in ETM are pre-trained. Word embeddings here are the Chinese word embeddings pre-trained by Tencent.

(viii) **Author Topic Model (ATM) [64]:** This model infers topic distributions specific over document to each user, and we take the top-k words with largest generation probability over all the topics of each author as his profiling words. We set $\alpha = 0.1$, $\beta = 0.01$ as the ATM priors, and $T = 50$ as the numbers of topics.

(ix) **Co-embedding Model (CEM) [47]:** This is the variant of our model, and it is the same as our proposed FCCM model except that it co-embeds users and words by utilizing our proposed variational auto-encoder but without applying the additional constraints.

(x) **Constrained Co-embedding Model (CCEM) [47]:** This is the variant of our model, and it is the same as our proposed FCCM model except that it co-embeds users and words by utilizing our variational auto-encoder with additional constraints but without adopting normalizing flow.

As a default setting in both our FCCM model and the baseline models, we set the size of the dimensions of the semantic representations to be 200, which is aligned to that of the settings in many embedding model such as the pre-trained Chinese word embedding model [66]. In terms of the Word2Vec model, we set the window size to be 5. To be consistent, we also set the window size to be 5 when we take into account co-occurrence information between any two words, i.e., assume the two words co-occur if they appear within a window with size being 5, and build corresponding word-to-word co-occurrence matrix, i.e., $\mathbf{W}$ as the input of our FCCM and the variant of our models, CEM and CCEM.

6.4 Evaluation Metrics

For evaluation purpose, a number of traditional relevance-oriented evaluation metrics are used: R-Prec (R-precision), MAP (Mean Average Precision), Pre@k (Precision at k), NDCG@k (Normalized Discounted Cumulative Gain at k), MRR@k (Mean Reciprocal Rank at k) [9]. 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. As our model and the baseline models work with inferred representations, we also use the *semantic* variants of the standard metrics, denoting as R-Prec-S, MAP-S, Pre-S@k, NDCG-S@k, and MRR-S@k, respectively. The standard metrics and the corresponding semantic ones differ between them in the way to obtain the relevance score of a retrieval keyword w^* and the keyword in the ground truth w^{gt} for user profiling. Specifically, for standard metrics, we let the relevance score be 1 if $w^* = w^{gt}$, otherwise be 0, whereas we let the relevance score be calculated by cosine similarity between the word embeddings of these two keywords w^* and w^{gt} in semantic version. For each metric we let k be 5, 10 and 20 (for MRR and MRR-S, k is set to 1, 3 and 5) to compute $M@k$, where M is one of the metrics.

7 RESULTS AND ANALYSIS

In this section, we answer the research questions listed in § 6.1, report and analyze the experimental results.

7.1 Overall Performance (RQ1)

To answer RQ1, we conduct experiments on the user profiling task, and compare the performance of our FCCM with the baselines methods listed in §6.3. Table 4 shows the result of user profiling over

Table 4. Performance of FCCM 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 FCCM and the best baseline model are denoted using ▲at the upper right corner of the FCCM 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
NVDM	.228	.115	.098	.085	.067	.094	.088	.069	.037	.031	.039
ATM	.228	.117	.097	.088	.066	.098	.090	.071	.038	.037	.041
Word2Vec	.239	.133	.092	.090	.061	.101	.090	.073	.034	.036	.037
ETM	.263	.140	.133	.124	.101	.129	.120	.105	.039	.033	.035
CEM	.275	.154	.141	.130	.097	.143	.135	.110	.045	.046	.051
Labeled ETM	.283	.166	.155	.140	.112	.144	.135	.122	.048	.044	.051
CCEM	.328	.194▲	.188	.184▲	.121	.190▲	.186▲	.141	.057	.059	.067
FCCM	.336▲	.185	.197▲	.183	.140▲	.188	.181	.155▲	.062▲	.066▲	.068▲

FCCM and the baseline methods in terms of the standard evaluation metrics discussed in §6.4. As a sanity check, we can observe from Table 4 that, the performance ranking across the comparison models is consistent over different evaluation metrics. More precisely, we can obtain the following performance ranking in most cases: FCCM ~ CCEM > Labeled ETM > CEM > ETM > Word2Vec > ATM > NVDM ~ TWE > AWE > TF-IDF, where we let > denote that the performance difference is statistically significant at a significance level of 95% by the Student's two tailed t-test, and ~ denotes that the difference is not statistically significant. CCEM and our FCCM can consistently perform better for the user profiling task than the other baseline methods. In addition, we also obtain the evaluation performances in terms of the semantic evaluation metrics, which are shown in Table 5. We can see that our proposed FCCM model significantly outperforms all other baseline models. In particular, we can observe the following performance ranking: FCCM ~ CCEM > CEM ~ Labeled ETM > ETM > Word2Vec > ATM > TWE ~ NVDM > AWE > TF-IDF. We can find that our proposed models, i.e., FCCM and CCEM, outperform other state-of-the-art models, indicating that integrating the voting information in QACs and co-embedding users and words into the same semantic space can significantly improve the performance of user profiling task. FCCM works slightly better than CCEM on the semantic version of standard metrics. This is due to the reason that user embeddings and word embeddings produced by FCCM are more suitable for the current dataset, thereby enhancing the performance of the user profiling task.

In order to validate the generalization ability of our proposed model, we also compare our model with the baseline models over the 5 subsets of our dataset. Table 6 and Table 7 show the comparison result of relevance-oriented metrics (NDCG@k) and the semantic-oriented metrics (NDCG-S@k) on user groups of #1 #2 and #3, while for brevity the comparison results of #4 and #5 and over other metrics (such as MAP and MAP-S) are omit here since they show the same trend and result with the some conclusion. As we can see in both Table 6 and Table 7 , the results over different user groups show the same performance ranking and are consistent with the result of the full Zhihu dataset. This result suggest that our model is able to generalize to different types of datasets and be adapted into the users over different topics.

Table 5. Profiling performance of FCCM 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 FCCM and the best baseline model are the same as those in Table 4.

			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
NVDM	.590	.678	.592	.585	.560	.625	.618	.608	.275	.264	.267
TWE	.593	.683	.601	.591	.563	.622	.613	.588	.268	.257	.273
ATM	.599	.701	.603	.598	.563	.624	.616	.605	.275	.282	.298
Word2Vec	.605	.723	.607	.595	.568	.684	.679	.652	.290	.289	.307
ETM	.613	.720	.618	.610	.595	.710	.699	.687	.301	.283	.294
Labeled ETM	.640	.761	.635	.618	.601	.723	.715	.695	.306	.289	.295
CEM	.622	.726	.627	.619	.589	.715	.704	.684	.294	.280	.299
CCEM	.659	.774▲	.647	.641	.606	.757▲	.743	.710	.336	.324	.338▲
FCCM	.667▲	.768	.651▲	.645▲	.623▲	.751	.747▲	.720▲	.341▲	.333▲	.336

Table 6. Performance comparison of FCCM and the baselines over the subsets in terms of the relevance-oriented metrics NDCG@k. The statistical significance is tested using a two-tailed paired t-test. Statistically significant differences between FCCM and the best baseline model are denoted using ▲at the upper right corner of the FCCM performance scores.

	Group #1			Group #2			Group #3		
NDCG@	5	10	20	5	10	20	5	10	20
TF-IDF	.092	.085	.072	.090	.089	.073	.086	.081	.066
AWE	.107	.105	.085	.103	.103	.085	.102	.096	.078
TWE	.117	.112	.097	.119	.111	.093	.111	.109	.091
NVDM	.125	.127	.104	.128	.122	.100	.120	.119	.096
ATM	.134	.134	.111	.136	.132	.112	.129	.127	.107
Word2Vec	.146	.139	.121	.142	.140	.119	.140	.134	.118
ETM	.156	.151	.128	.154	.147	.126	.147	.144	.124
CEM	.158	.151	.133	.159	.153	.130	.151	.149	.129
Labeled ETM	.164	.160	.139	.160	.145	.137	.157	.152	.133
CCEM	.173	.170	.146	.175	.166	.150	.167	.162	.141
FCCM	.177▲	.174▲	.149▲	.177 ▲	.168 ▲	.150	.171▲	.167▲	.148▲

7.2 Impact of Constraints on the Profiling Performance (RQ2)

Next, we make a comparison on the profiling performance of FCCM, CCEM and other two best baselines, CEM and Labled ETM to examine the effectiveness of the integrated constraints.

Recall that in both FCCM and CCEM models, constraints of the representations of users and words are integrated into the the objective, while there are no such constraints integrated in other representative baseline models, i.e., CEM and Labled ETM. Fig. 4 provides the user profiling results evaluated by the semantic metrics, i.e., Pre-S@5, NDCG-S@5, MRR-S@1, MAP-S. In Fig. 4, we vary the trade-off parameter α from 0 to 0.003 with a step size of 0.0005. In the figure, $\alpha = 0$ indicates that the constraints are not applied in both FCCM and CCEM, while a larger α means that more

Table 7. Performance comparison of FCCM and the baselines over the subsets in terms of the semantic-oriented metrics NDCG-S@k. The statistical significance is tested using a two-tailed paired t-test. Statistically significant differences between FCCM and the best baseline model are denoted using ▲at the upper right corner of the FCCM performance scores.

	Group #1			Group #2			Group #3		
NDCG-S@	5	10	20	5	10	20	5	10	20
TF-IDF	.516	.510	.484	.513	.510	.483	.505	.503	.476
AWE	.545	.539	.508	.547	.541	.507	.541	.535	.501
TWE	.577	.572	.539	.573	.571	.536	.569	.565	.531
NVDM	.593	.607	.560	.594	.603	.564	.590	.600	.559
ATM	.607	.633	.594	.610	.630	.595	.605	.625	.589
Word2Vec	.636	.648	.627	.638	.648	.625	.631	.644	.620
ETM	.658	.662	.643	.658	.662	.642	.653	.657	.633
CEM	.675	.677	.667	.676	.680	.666	.673	.674	.658
Labeled ETM	.695	.689	.682	.695	.686	.681	.690	.681	.675
CCEM	.714	.707	.697	.715	.710	.697	.707	.702	.695
FCCM	.722▲	.725▲	.706▲	.720▲	.724▲	.712▲	.716▲	.721▲	.702▲

weights are put on the constraints. As shown in the figure, over all the α values that are larger than 0.0005, both our FCCM and CCEM models that integrate the voting constraints, perform clearly better than the CEM and the Labeled ETM models, and both achieve their best performances at fix α value ($\alpha = 0.0015$), which confirms the fact that integrating constraints into the model indeed helps to improve the performance. A detailed analysis of the parameter α can be found in §7.6.

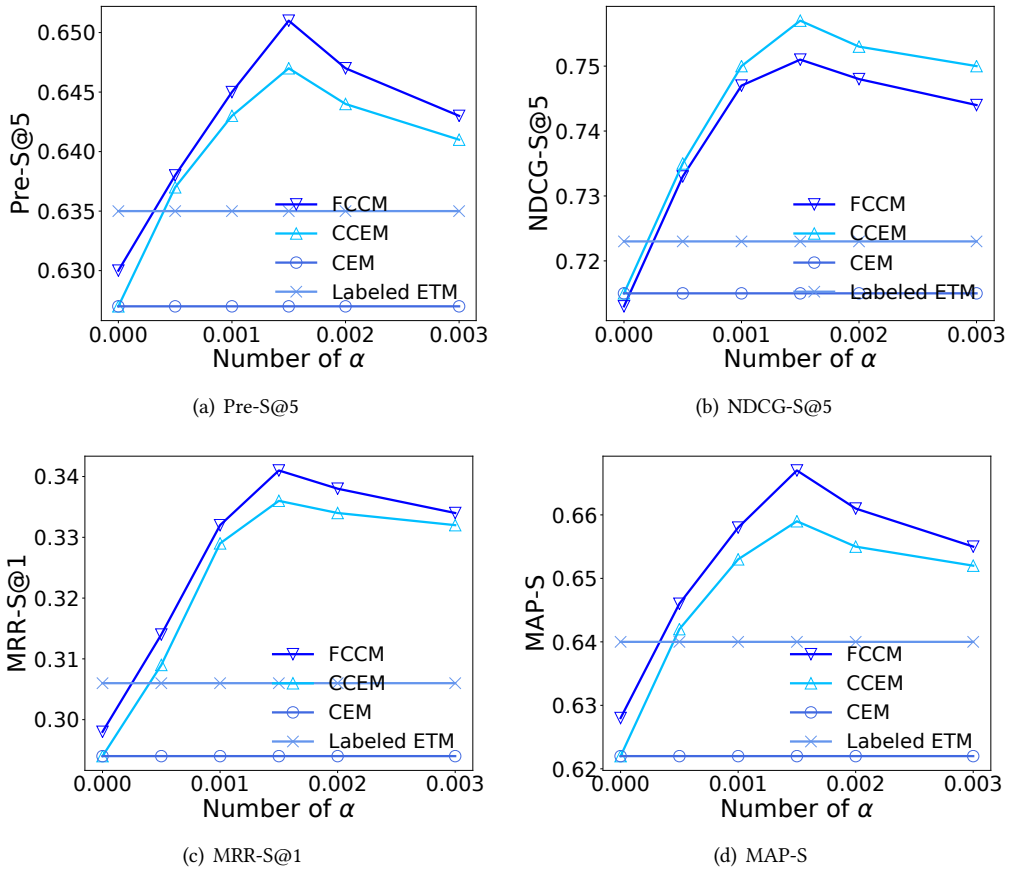
To conclude for RQ2, we have shown that by varying the contribution weights of the voting constraints, our model witnessed different user profiling performances and best performances are achieve at a fix weight ($\alpha = 0.0015$), which validates that additional voting information modelled by our constraints can indeed boost the user profiling performance.

7.3 Effect of Co-embedding Users and Words (RQ3)

To further validate the effectiveness of the co-embedding approach used in our model, we make a user profiling performance comparison between the co-embedding models that co-embed users and words into the same semantic space and the non-co-embedding ones. From Table 5, we can observe that those co-embedding methods, i.e., FCCM, CCEM, CEM, and ETM, significantly perform better than the other non-co-embedding baselines. This result 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 FCCM and CEM significantly contribute to the improvement of user profiling performance. Note that in the original input space, the user representation and word representation are clearly in different embedding spaces, but our FCCM can effectively measure the similarities be the embeddings of the two entities for the user profiling task, which answers RQ3.

7.4 Effect of Normalizing flow (RQ4)

Now we turn to RQ4 to figure out how the normalizing flow affects the performance. We make a comparison on the user profiling performance of FCCM and CCEM (i.e., our FCCM without normalizing flow) based on the result shown Table 4. We can see that the flow-based method, i.e., our FCCM, general perform better than the CCEM model on all the standard evaluation metrics.



(a) Pre-S@5 (b) NDCG-S@5

(c) MRR-S@1 (d) MAP-S

Fig. 4. User profiling performance comparisons among FCCM, CCEM, and other two best baselines with different α evaluated by Pre-S@5, NDCG-S@5, MRR-S@1 and MAP-S.

However, we believe the semantic evaluation metrics are more reasonable than the standard evaluation metrics for the user profiling task. Specifically, it can be observed in Table 5 that FCCM works better than CCEM on almost all of the semantic evaluation metrics except for R-Prec-S, NDCG-S@5, and MRR-S@5. This result indicates that our proposed flow-based method can effectively transform the initial variational posteriors to approximate the true posteriors, such that the embeddings produced by FCCM can significantly contribute to the improvement of user profiling performance. This result validates the effectiveness of the normalizing flow approach. Indeed, our FCCM model is mostly based on the Variational Auto-Encoder model and the variational inference framework. The choice of approximate posterior distribution is one of the core problems in variational inference. Most applications of variational inference employ simple families of posterior approximations in order to allow for efficient inference, such as Gaussian distributions that are used in our CCEM, focusing on mean-field or other simple structured approximations. This restriction has a significant impact on the quality of variational inferences. Especially in the representations of multiple entities, the semantic representations are normally discrete or more complex than those in the space of the Gaussian distributions. The "normalizing flow" approach is able to specify flexible,

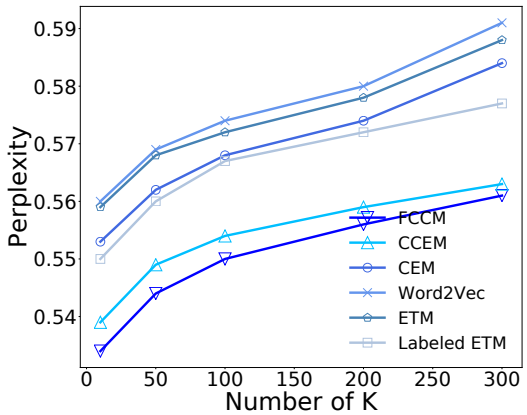


Fig. 5. Perplexity performance of FCCM and the baselines with different size of top-K keywords set.

arbitrarily complex and scalable approximate posterior distributions, and thereafter provides better performance over the original models [35, 62].

7.5 Quality of Semantic Representations (RQ5)

In order to answer RQ5, we evaluate the performance of FCCM and the baseline models by the perplexity, which is a widely used metric to evaluate the generation of representations [4, 44] and is often used to evaluate the performance of topic models [4]. The perplexity value is monotonically decreasing with the likelihood of documents, and is algebraically equivalent to the inverse of the geometric mean per-word likelihood. In our user profiling task, to evaluate the quality of both representation of users and words, we modify the calculation of perplexity to be: $Perplexity(\mathcal{K}_u) = \exp\{-\frac{\sum_{i=0}^{|\mathcal{K}_u|} \log p(w_{u,i}|u)}{|\mathcal{K}_u|}\}$, 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. With such measure, a lower perplexity score indicates better generalization performance. Fig. 5 shows the mean perplexity performances of FCCM and the baselines models, over different size of top-K keywords set, i.e., different number of K, ranging form 10 to 300. We can see that our FCCM consistently perform better than other baselines models on the perplexity metric over all the K values, which demonstrates the high quality of the embeddings of both users and words produced by FCCM.

7.6 Impact of the Trade-off Parameters (RQ6)

As we mentioned in §4.4 and §5.1, our model contains two trade-off parameters, namely α balancing the ELBO and voting constraints (see Eq. (28)) and β balancing the reconstruction weights between the words and the users (see Eq. (6)). It is nature to ask how the different values of each trade-off parameters impact the user profiling performance in our model (i.e. RQ6).

To answer this question, we first fix the parameter β to be optimal (i.e. $\beta = 1.5$ for both FCCM and CCEM), vary the parameter α from 0 to 0.003 with a step size of 0.0005 and obtain the performance of our model under different values. The result is plotted in Fig. 4. We can see that when α increases from 0 to 0.0015 the user profiling performances observe a boosting trend in both our FCCM and CCEM models, and both the baseline models, i.e., CEM and the Labeled ETM, perform worse than FCCM and CCEM with $\alpha = 0.0015$, demonstrating that integrating the constraints in our proposed embedding model indeed affect the user profiling performance and by tuning the contribution of the

voting constraints we can obtain considerable improvements. In particular, when α increasing from 0.0015 to 0.003, the profiling performances of both both FCCM and CCEM decrease. The reason behind this is that when α is set to be larger, FCCM would ignore the reconstruction of words and users during the process of the model optimization, gradually losing the ability of capturing semantic affinities between words and users.

We then turn to examine the impact of the trade-off parameter α that governs the weights between embeddings of users and words. We fix the parameter α to be optimal, vary the parameter β from 0.5 to 2.5 with a step size of 0.1 for our FCCM, CCEM, and CEM models. The comparison result are shown in Fig. 6. We can observe that the user profiling performances of all the three co-embedding models, i.e. FCCM, CCEM and CEM, increases when varying β from 0.5 to 1.5, in terms of the evaluation metrics Pre-S@5, NDCG-S@5, MRR-S@1, and MAP-S. The performance of FCCM and the baseline models observes a slight decrease when β continue to increase from 1.6. This result indicates that enforcing the trade-off parameter α to govern the weights between embeddings of users and words is necessary to obtain a better user profiling performance in QACs.

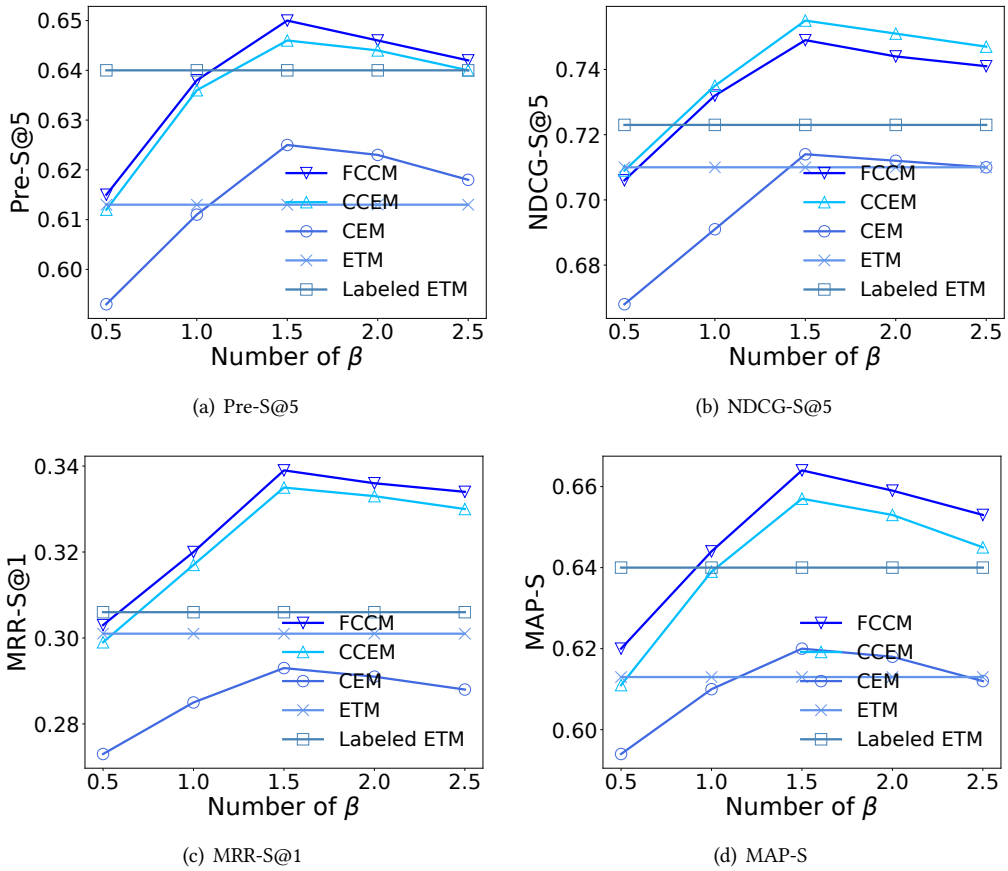
In addition, as can be seen in both Fig. 4 and Fig. 6, in most cases FCCM outperforms the baseline models in terms of all the evaluation metrics with a variety of α and β, which demonstrates the robustness and superiority of our proposed FCCM model in which additional constraints and normalizing flows are integrated.

7.7 Dimensions of the Embeddings (RQ7)

Finally, we study the impact of the embedding dimensions to the user profiling performances of the tested models. Specifically, we vary the dimensions of the inferred embeddings of users and words, ranging from 50 to 400, for all the tested models (i.e. our FCCM model and the baseline models), and report the performances of them over the Pre-S@5 and NDCG-S@5 metrics. Fig. 7 shows the comparison result. In general, we find that the evaluation results can be observed essentially the same trend over these two representative metrics: the more dimension used, the better performances can be obtained. We also can see that the user profiling performance in QACs significantly increases in both our FCCM and the baseline models when the sizes of the dimensions increase from 50 to ~250, while after 250, all the performance seems to reach a plateau as the dimensions increase to 400. In most cases with different sizes of the inferred embeddings, our FCCM model consistently outperforms the baseline models in terms of the both metrics, which further validates the effectiveness of our proposed model. To conclude with Fig. 7, we demonstrate that the performance improvement of FCCM over the baselines is not sensitive to the sizes of the inferred dimensions of users and words.

8 CONCLUSION

In this paper, we have studied the task of user profiling in Question Answering Communities: given a set of users, the questions asked by the users, the answers associated with the questions, and the auxiliary information such as thumbs-up and thumbs-down on the answers in QACs, retrieve a rank list of keywords to profile a user such that the keywords can summarize the user's broad expertise. To address the task, we propose a novel Flow-based Constrained Co-embedding Model, abbreviated as FCCM. Our FCCM model jointly co-embeds the embedding representations of users and words in the same semantic space such that the semantic affinities between them can be effectively measured for generating sets of keywords to profile the users in QACs. To achieve the goal of jointly inferring the representations of users and words, our FCCM extends the standard variational auto-encoder algorithm, which consists of a neural encoder that maps the input users and words into flexible distributions and a neural decoder that effectively reconstructs the input

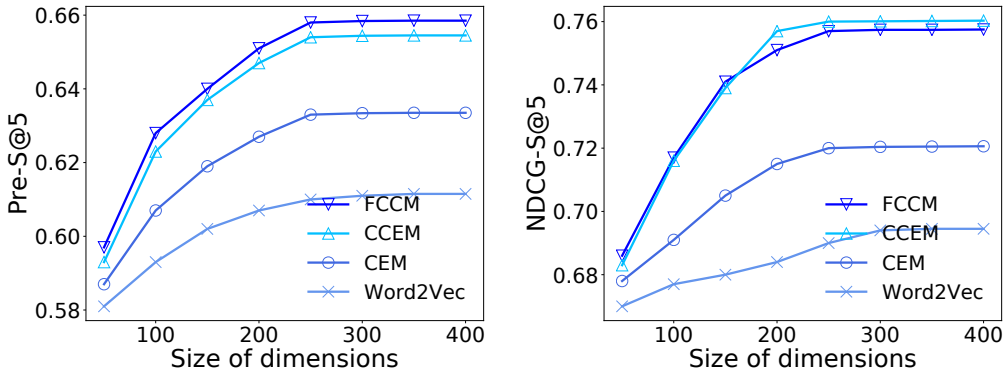


(a) Pre-S@5 (b) NDCG-S@5

(c) MRR-S@1 (d) MAP-S

Fig. 6. User profiling performance comparisons among FCCM, CCEM, CEM, and the other baselines with different β evaluated by Pre-S@5, NDCG-S@5, MRR-S@1 and MAP-S.

users and words based on the input inferred representations of users and words. Unlike all the previous user profiling algorithms, our FCCM model fully takes into account the unique auxiliary information in QACs. In particular, the auxiliary information, i.e., the thumbs-up and thumbs-down of answers, is utilized to construct the voting constraints in our FCCM model to enforce the inferred embeddings subject to these constraints. Unlike traditional variational auto-encoder algorithms that assume the distributions as Gaussian, our FCCM improve the performance by integrating normalizing flow to transform the Gaussian in traditional VAEs to a more flexible function that applies a sequence of invertible and smooth mappings from the input to the output. To effectively infer the final embeddings of users and words, we propose an inference algorithm for our FCCM model. Our inference algorithm iteratively update the objective to obtain optimal parameters such that the variational latent variables subject to the voting constraints while still able to co-embed users and words into the same semantic space. Through our proposed FCCM model to address user profiling task in QACs as an example, we would provide inspiration and insight on how to extend standard VAEs to address other tasks in information retrieval, where semantic vector representations of multiple categories of entities need to be inferred in the same semantic



(a) Pre-S@5 (b) NDCG-S@5

Fig. 7. Pre-S@5 and NDCG-S@5 performance of FCCM and the baselines with various sizes of dimensions of the inferred embeddings.

space. Experiments conducted on the Chinese Zhihu dataset demonstrate the effectiveness of the proposed FCCM model.

As to future work, there are many unexplored avenues. For instance, can we integrate more additional auxiliary information such as users' social communities, timestamps of the posted questions and answers, and location information, to boost the performance of user profiling task in QACs? Can we address the task of user profiling in QACs by extending other state-of-the-art neural network techniques such as disentanglement embedding [63]? In particular, our flow-based constrained co-embedding models can also be applied to other domains. For instance, it can be used into the recommendation systems to jointly learn the embeddings of users and items with textual side information, which is also a good direction to explore.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (Grant No. 61906219). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Krisztian Balog, Toine Bogers, Leif Azzopardi, Maarten De Rijke, and Antal Van Den Bosch. 2007. Broad expertise retrieval in sparse data environments. In *Proceedings of SIGIR*. 551–558.

[2] Krisztian Balog, Maarten De Rijke, et al. 2007. Determining Expert Profiles (With an Application to Expert Finding).. In *Proceedings of IJCAI*, Vol. 7. 2657–2662.

[3] Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. 2018. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649* (2018).

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

[5] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *Proceedings of CONLL* (2015).

[6] Weijian Chen, Yulong Gu, Zhaochun Ren, Xiangnan He, Hongtao Xie, Tong Guo, Dawei Yin, and Yongdong Zhang. [n. d.]. Semi-supervised User Profiling with Heterogeneous Graph Attention Networks.

[7] Yifan Chen, Yang Wang, Xiang Zhao, Hongzhi Yin, Ilya Markov, and MAARTEN De Rijke. [n. d.]. Local Variational Feature-Based Similarity Models for Recommending Top-N New Items. *ACM Transactions on Information Systems* 38, 2, Article 12 (Feb. [n. d.]), 33 pages.

[8] Nick Craswell, Arjen P de Vries, and Ian Soboroff. 2005. Overview of the TREC 2005 Enterprise Track.. In *Trec*, Vol. 5. 1–7.
[9] W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search engines: Information retrieval in practice*. Vol. 520. Addison-Wesley Reading.
[10] Ali Daud. 2012. Using time topic modeling for semantics-based dynamic research interest finding. *Knowledge-Based Systems* 26 (2012), 154–163.
[11] Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. 2018. Hyperspherical variational auto-encoders. In *Proceedings of UAI*. 856–865.
[12] Maarten De Rijke, Krisztian Balog, Toine Bogers, and Antal Van Den Bosch. 2010. On the evaluation of entity profiles. In *CLEF*. 94–99.
[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[14] Adji B Dieng, Francisco JR Ruiz, and David M Blei. 2019. Topic modeling in embedding spaces. *arXiv preprint arXiv:1907.04907* (2019).
[15] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).
[16] Alexey Dosovitskiy and Thomas Brox. 2016. Generating images with perceptual similarity metrics based on deep networks. In *Advances in NeurIPS*. 658–666.
[17] Hui Fang and ChengXiang Zhai. 2007. Probabilistic models for expert finding. In *Proceedings of ECIR*. 418–430.
[18] Yi Fang and Archana Godavarthy. 2014. Modeling the dynamics of personal expertise. In *Proceedings of SIGIR*. 1107–1110.
[19] Golnoosh Farnadi, Lise Getoor, Marie-Francine Moens, and Martine De Cock. 2020. User Profiling Using Hinge-loss Markov Random Fields. *arXiv preprint arXiv:2001.01177* (2020).
[20] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. 2015. Made: Masked autoencoder for distribution estimation. In *Proceedings of ICML*. 881–889.
[21] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of ASC*. 315–323.
[22] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of SIGKDD*. 855–864.
[23] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, and Dawei Yin. 2020. Hierarchical User Profiling for E-commerce Recommender Systems. In *Proceedings of WSDM*. 223–231.
[24] Ralf Herbrich. 2000. Large margin rank boundaries for ordinal regression. *Advances in large margin classifiers* (2000), 115–132.
[25] Emiel Hoogeboom, Rianne van den Berg, and Max Welling. 2019. Emerging convolutions for generative normalizing flows. *arXiv preprint arXiv:1901.11137* (2019).
[26] Renjun Hu, Charu C Aggarwal, Shuai Ma, and Jinpeng Huai. 2016. An embedding approach to anomaly detection. In *Proceedings of ICDE*. 385–396.
[27] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *Proceedings of WSDM*. 731–739.
[28] Shuhui Jiang, Xueming Qian, Jialie Shen, Yun Fu, and Tao Mei. 2015. Author topic model-based collaborative filtering for personalized POI recommendations. *IEEE Transactions on Multimedia* 17, 6 (2015), 907–918.
[29] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).
[30] Wei-Chen Kao, Duen-Ren Liu, and Shiu-Wen Wang. 2010. Expert finding in question-answering websites: a novel hybrid approach. In *Proceedings of SAC*. 867–871.
[31] Maryam Karimzadehgan, Ryen W White, and Matthew Richardson. 2009. Enhancing expert finding using organizational hierarchies. In *Proceedings of ECIR*. 177–188.
[32] Youngwoo Kim, Myungha Jang, and James Allan. 2020. Explaining Text Matching on Neural Natural Language Inference. *ACM Transactions on Information Systems* 38, 4 (2020), 1–23.
[33] Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in NeurIPS*. 10215–10224.
[34] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in NeurIPS*. 3581–3589.
[35] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in NeurIPS*. 4743–4751.
[36] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *Proceedings of ICLR* (2013).

[37] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[38] Rémi Lebret and Ronan Collobert. 2013. Word emdeddings through hellinger PCA. *Proceedings of EACL* (2013).

[39] Jiwei Li, Alan Ritter, and Eduard Hovy. 2014. Weakly supervised user profile extraction from twitter. In *Proceedings of ACL*. 165–174.

[40] Rui Li, Chi Wang, and Kevin Chen-Chuan Chang. 2014. User profiling in an ego network: co-profiling attributes and relationships. In *Proceedings of WWW*. 819–830.

[41] Shangsong Liang. 2018. Dynamic user profiling for streams of short texts. In *Proceedings of AAAI*.

[42] Shangsong Liang. 2019. Collaborative, dynamic and diversified user profiling. In *Proceedings of AAAI*.

[43] Shangsong Liang and Maarten de Rijke. 2016. Formal language models for finding groups of experts. *Information Processing & Management* 52, 4 (2016), 529–549.

[44] Shangsong Liang, Emine Yilmaz, and Evangelos Kanoulas. 2016. Dynamic clustering of streaming short documents. In *Proceedings of SIGKDD*. 995–1004.

[45] S. Liang, E. Yilmaz, and E. Kanoulas. 2019. Collaboratively Tracking Interests for User Clustering in Streams of Short Texts. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2019), 257–272.

[46] Shangsong Liang, Xiangliang Zhang, Zhaochun Ren, and Evangelos Kanoulas. 2018. Dynamic embeddings for user profiling in twitter. In *Proceedings of SIGKDD*. 1764–1773.

[47] Yupeng Luo, Shangsong Liang, and Zaiqiao Meng. 2019. Constrained Co-embedding Model for User Profiling in Question Answering Communities. In *Proceedings of CIKM*. 439–448.

[48] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. 2016. Predict Anchor Links across Social Networks via an Embedding Approach.. In *Proceedings of IJCAI*, Vol. 16. 1823–1829.

[49] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-embedding Attributed Networks. In *Proceedings of WSDM*. 393–401.

[50] Zaiqiao Meng, Shangsong Liang, Jinyuan Fang, and Teng Xiao. 2019. Semi-supervisedly co-embedding attributed networks. In *Advances in NeurIPS*. 6507–6516.

[51] Zaiqiao Meng, Shangsong Liang, Xiangliang Zhang, Richard McCreadie, and Iadh Ounis. 2020. Jointly Learning Representations of Nodes and Attributes for Attributed Networks. *ACM Transactions on Information Systems* 38, 2 (2020), 1–32.

[52] Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of ICML*. 1727–1736.

[53] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of ICLR* (2013).

[54] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in NeurIPS*. 3111–3119.

[55] George Papamakarios, Theo Pavlakou, and Iain Murray. 2017. Masked autoregressive flow for density estimation. In *Advances in NeurIPS*. 2338–2347.

[56] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*. 1532–1543.

[57] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of SIGKDD*. 701–710.

[58] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Proceedings of NAACL* (2018).

[59] Desislava Petkova and W Bruce Croft. 2008. Hierarchical language models for expert finding in enterprise corpora. *International Journal on Artificial Intelligence Tools* 17, 01 (2008), 5–18.

[60] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. 2019. Waveglow: A flow-based generative network for speech synthesis. In *Proceedings of ICASSP*. 3617–3621.

[61] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *arxiv*.

[62] Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770* (2015).

[63] Karl Ridgeway and Michael C Mozer. 2018. Learning deep disentangled embeddings with the f-statistic loss. In *Advances in NeurIPS*. 185–194.

[64] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. 487–494.

[65] Jan Rybak, Krisztian Balog, and Kjetil Nørvåg. 2014. Temporal expertise profiling. In *Proceedings of ECIR*. 540–546.

[66] Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional Skip-Gram: Explicitly Distinguishing Left and Right Context for Word Embeddings. In *Proceedings of NAACL*. 175–180.
[67] Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast Online Training with Frequency-Adaptive Learning Rates for Chinese Word Segmentation and New Word Detection. In *Proceedings of ACL*. 253–262.
[68] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of WWW*. 1067–1077.
[69] Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, and Ben Poole. 2019. Discrete Flows: Invertible Generative Models of Discrete Data. *arXiv preprint arXiv:1905.10347* (2019).
[70] Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, Maosong Sun, et al. 2016. Max-margin deepwalk: Discriminative learning of network representation.. In *Proceedings of IJCAI*, Vol. 2016. 3889–3895.
[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in NeurIPS*. 5998–6008.
[72] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.
[73] Joshua F Wiley. 2016. *R Deep Learning Essentials*. Packt Publishing Ltd.
[74] Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, and Xueqi Cheng. 2020. Dual-factor Generation Model for Conversation. *ACM Transactions on Information Systems* 38, 3 (2020), 1–31.
[75] Yukun Zhao, Shangsong Liang, Zhaochun Ren, Jun Ma, Emine Yilmaz, and Maarten de Rijke. 2016. Explainable user clustering in short text streams. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 155–164.

A DERIVATION OF ELBO

Substituting (4) and (5) into (3), we obtain the derivation of the ELBO in our model as follows:

$$
\begin{aligned}
&\mathbb{E}_{q_\phi}\left[\log \frac{p(\mathbf{W},\mathbf{V},\mathbf{Y},\mathbf{Z})}{q_\phi(\mathbf{Y},\mathbf{Z}\mid\mathbf{W},\mathbf{V})}\right] \\
&= \mathbb{E}_{q_\phi}\left[\log \frac{p(\mathbf{Y})p(\mathbf{Z})\prod_{(i,j)\in\mathcal{E}_\mathcal{W}} p_{\theta_1}(\mathbf{W}_{ij}\mid\mathbf{Y}_i,\mathbf{Y}_j)\prod_{(u,i)\in\mathcal{E}_\mathcal{U}} p_{\theta_2}(\mathbf{V}_{ui}\mid\mathbf{Z}_u,\mathbf{Y}_i)}{\prod_{i\in\mathcal{W}} q_{\phi_1}(\mathbf{Y}_i\mid\mathbf{W})\prod_{u\in\mathcal{U}} q_{\phi_2}(\mathbf{Z}_u\mid\mathbf{V})}\right] \\
&= \mathbb{E}_{q_\phi}\left[\sum_{(i,j)\in\mathcal{E}_\mathcal{W}} \log p_{\theta_1}(\mathbf{W}_{ij}\mid\mathbf{Y}_i,\mathbf{Y}_j)\right] + \mathbb{E}_{q_\phi}\left[\sum_{(u,i)\in\mathcal{E}_\mathcal{U}} \log p_{\theta_2}(\mathbf{V}_{ui}\mid\mathbf{Z}_u,\mathbf{Y}_i)\right] \\
&+ \mathbb{E}_{q_\phi}\left[\log \frac{p(\mathbf{Y})p(\mathbf{Z})}{\prod_{i\in\mathcal{W}} q_{\phi_1}(\mathbf{Y}_i\mid\mathbf{W})\prod_{u\in\mathcal{U}} q_{\phi_2}(\mathbf{Z}_u\mid\mathbf{V})}\right] \\
&= \mathbb{E}_{q_\phi}\left[\sum_{(i,j)\in\mathcal{E}_\mathcal{W}} \log p_{\theta_1}(\mathbf{W}_{ij}\mid\mathbf{Y}_i,\mathbf{Y}_j)\right] + \mathbb{E}_{q_\phi}\left[\sum_{(u,i)\in\mathcal{E}_\mathcal{U}} \log p_{\theta_2}(\mathbf{V}_{ui}\mid\mathbf{Z}_u,\mathbf{Y}_i)\right] \\
&- \mathbb{E}_{q_\phi}\left[\log \frac{\prod_{i\in\mathcal{W}} q_{\phi_1}(\mathbf{Y}_i\mid\mathbf{W})}{p(\mathbf{Y})} + \log \frac{\prod_{u\in\mathcal{U}} q_{\phi_2}(\mathbf{Z}_u\mid\mathbf{V})}{p(\mathbf{Z})}\right] \\
&= \mathbb{E}_{q_\phi}\left[\sum_{(i,j)\in\mathcal{E}_\mathcal{W}} \log p_{\theta_1}(\mathbf{W}_{ij}\mid\mathbf{Y}_i,\mathbf{Y}_j)\right] + \mathbb{E}_{q_\phi}\left[\sum_{(u,i)\in\mathcal{E}_\mathcal{U}} \log p_{\theta_2}(\mathbf{V}_{ui}\mid\mathbf{Z}_u,\mathbf{Y}_i)\right] \\
&- D_{KL}(q_{\phi_1}(\mathbf{Y}\mid\mathbf{W})\parallel p(\mathbf{Y})) - D_{KL}(q_{\phi_2}(\mathbf{Z}\mid\mathbf{V})\parallel p(\mathbf{Z})).
\end{aligned}
\tag{52}
$$

In order to enhance the performance of our proposed model for user profiling task, a trade-off parameter β is utilized to balance the optimization process of word embedding and user embedding. Thus, we have:

$$\begin{aligned}\log p(\mathbf{W},\mathbf{V}) &\geq \mathcal{L}(\theta_1,\phi_1;\mathbf{W}) + \beta\mathcal{L}(\theta_2,\phi_2;\mathbf{W},\mathbf{V}) \\ &\triangleq \mathcal{L}(\theta,\phi;\mathbf{W},\mathbf{V}),\end{aligned} \tag{53}$$

where

$$\begin{aligned}\mathcal{L}(\theta_1,\phi_1;\mathbf{W}) = \mathbb{E}_{q_\phi}&\left[\sum_{(i,j)\in\mathcal{E}_\mathcal{W}} \log p_{\theta_1}(\mathbf{W}_{ij} \mid \mathbf{Y}_i,\mathbf{Y}_j)\right] \\ &- D_{KL}(q_{\phi_1}(\mathbf{Y}\mid\mathbf{W}) \parallel p(\mathbf{Y})),\end{aligned} \tag{54}$$

$$\begin{aligned}\mathcal{L}(\theta_2,\phi_2;\mathbf{W},\mathbf{V}) = \mathbb{E}_{q_\phi}&\left[\sum_{(u,i)\in\mathcal{E}_\mathcal{U}} \log p_{\theta_2}(\mathbf{V}_{ui} \mid \mathbf{Z}_u,\mathbf{Y}_i)\right] \\ &- D_{KL}(q_{\phi_2}(\mathbf{Z}\mid\mathbf{V}) \parallel p(\mathbf{Z})).\end{aligned} \tag{55}$$

B CALCULATION OF LOGARITHM OF VARIATIONAL POSTERIOR

We aim to obtain the value of $\log q_{\phi_1}(\mathbf{Y}_i^{(T)} \mid \mathbf{W})$ and $\log q_{\phi_2}(\mathbf{Z}_u^{(T)} \mid \mathbf{V})$. According the rule for transformation of densities [62], we have the recursive relationship as follows:

$$\begin{aligned}q_{\phi_1}\left(\mathbf{Y}_i^{(t)}|\mathbf{W}\right) &= q_{\phi_1}\left(\mathbf{Y}_i^{(t-1)}|\mathbf{W}\right)\left|\frac{d\mathbf{Y}_{i-1}^{(t)}}{d\mathbf{Y}_i^{(t-1)}}\right|^{-1}, \\ q_{\phi_2}\left(\mathbf{Z}_u^{(t)}|\mathbf{V}\right) &= q_{\phi_2}\left(\mathbf{Z}_u^{(t-1)}|\mathbf{V}\right)\left|\frac{d\mathbf{Z}_u^{(t)}}{d\mathbf{Z}_u^{(t-1)}}\right|^{-1},\end{aligned} \tag{56}$$

where $|\cdot|$ represents the determinant of matrix and $\frac{d\mathbf{Y}_i^{(t)}}{d\mathbf{Y}_i^{(t-1)}}$ and $\frac{d\mathbf{Z}_u^{(t)}}{d\mathbf{Z}_u^{(t-1)}}$ are Jacobin matrices. Solving the recursive formula (56), we obtain:

$$\begin{aligned}q_{\phi_1}\left(\mathbf{Y}_i^{(T)}|\mathbf{W}\right) &= q_{\phi_1}\left(\mathbf{Y}_i^{(0)}|\mathbf{W}\right)\prod_{t=1}^{T}\left|\frac{d\mathbf{Y}_{i-1}^{(t)}}{d\mathbf{Y}_i^{(t-1)}}\right|^{-1}, \\ q_{\phi_2}\left(\mathbf{Z}_u^{(T)}|\mathbf{V}\right) &= q_{\phi_2}\left(\mathbf{Z}_u^{(0)}|\mathbf{V}\right)\prod_{t=1}^{T}\left|\frac{d\mathbf{Z}_u^{(t)}}{d\mathbf{Z}_u^{(t-1)}}\right|^{-1}.\end{aligned} \tag{57}$$

Finally, we obtain:

$$\begin{aligned}\log q_{\phi_1}\left(\mathbf{Y}_i^{(T)}|\mathbf{W}\right) = \log q_{\phi_1}&\left(\mathbf{Y}_i^{(0)}|\mathbf{W}\right) \\ &- \sum_{t=1}^{T}\log\left|\frac{d\mathbf{Y}_i^{(t)}}{d\mathbf{Y}_i^{(t-1)}}\right|,\end{aligned} \tag{58}$$

$$\begin{aligned}\log q_{\phi_2}\left(\mathbf{Z}_u^{(T)}|\mathbf{V}\right) = \log q_{\phi_2}&\left(\mathbf{Z}_u^{(0)}|\mathbf{V}\right) \\ &- \sum_{t=1}^{T}\log\left|\frac{d\mathbf{Z}_u^{(t)}}{d\mathbf{Z}_u^{(t-1)}}\right|.\end{aligned} \tag{59}$$

C CALCULATION OF FINAL ELBO

The last two terms in (12) are the KL divergence terms:

$$KL = \mathbb{E}_{q_\phi}\left[\sum_{i\in\mathcal{W}} \log q_{\phi_1}(\mathbf{Y}_i^{(T)} \mid \mathbf{W}) - \log p_{\theta_1}(\mathbf{Y}_i)\right] + \mathbb{E}_{q_\phi}\left[\sum_{u\in\mathcal{U}} \log q_{\phi_2}(\mathbf{Z}_u^{(T)} \mid \mathbf{V}) - \log p_{\theta_2}(\mathbf{Z}_u)\right]. \tag{60}$$

Substituting (13) and (14) into (60), we have:

$$KL = \mathbb{E}_{q_\phi}\left[\sum_{i\in\mathcal{W}} \log q_{\phi_1}\left(\mathbf{Y}_i^{(0)}|\mathbf{W}\right) - \sum_{t=1}^{T}\log\left|\frac{d\mathbf{Y}_i^{(t)}}{d\mathbf{Y}_i^{(t-1)}}\right| - \log p_{\theta_1}\left(\mathbf{Y}_i^{(T)}\right)\right] + \mathbb{E}_{q_\phi}\left[\sum_{u\in\mathcal{U}} \log q_{\phi_2}\left(\mathbf{Z}_u^{(0)}|\mathbf{V}\right) - \sum_{t=1}^{T}\log\left|\frac{d\mathbf{Z}_u^{(t)}}{d\mathbf{Z}_u^{(t-1)}}\right| - \log p_{\theta_2}\left(\mathbf{Z}_u^{(T)}\right)\right]. \tag{61}$$

Substituting (20),(21),(18) and (19) into (61), we have:

$$KL = \mathbb{E}_{q_\phi}\left[\sum_{i\in\mathcal{W}} \log \mathcal{N}(\mathbf{Y}_i^{(0)}|\boldsymbol{\mu}_{\mathbf{Y}_i}, \boldsymbol{\sigma}^2_{\mathbf{Y}_i}\mathbf{I}) - \sum_{t=1}^{T}\log\left|\frac{d\mathbf{Y}_i^{(t)}}{d\mathbf{Y}_i^{(t-1)}}\right| - \log \mathcal{N}(\mathbf{Y}_i^{(T)}|\mathbf{0},\mathbf{I})\right] + \mathbb{E}_{q_\phi}\left[\sum_{u\in\mathcal{U}} \log \mathcal{N}(\mathbf{Z}_u^{(0)}|\boldsymbol{\mu}_{\mathbf{Z}_u}, \boldsymbol{\sigma}^2_{\mathbf{Z}_u}\mathbf{I}) - \sum_{t=1}^{T}\log\left|\frac{d\mathbf{Z}_u^{(t)}}{d\mathbf{Z}_u^{(t-1)}}\right| - \log \mathcal{N}(\mathbf{Z}_u^{(T)}|\mathbf{0},\mathbf{I})\right]. \tag{62}$$

Because all Gaussian distributions in (62) are diagonal Gaussian distribution, we have:

$$\begin{aligned}\mathcal{N}(\mathbf{Y}_i^{(0)}|\boldsymbol{\mu}_{\mathbf{Y}_i}, \boldsymbol{\sigma}^2_{\mathbf{Y}_i}\mathbf{I}) &= \left[(2\pi)^{\frac{D}{2}}\prod_{d=1}^{D}\boldsymbol{\sigma}_{\mathbf{Y}_i}^{(0)}\Big|_d\right]^{-1}\exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{\mathbf{Y}_i^{(0)}-\boldsymbol{\mu}_{\mathbf{Y}_i}^{(0)}}{\boldsymbol{\sigma}_{\mathbf{Y}_i}^{(0)}}\Big|_d\right)^2\right\}\\ &= \left[(2\pi)^{\frac{D}{2}}\prod_{d=1}^{D}\boldsymbol{\sigma}_{\mathbf{Y}_i}^{(0)}\Big|_d\right]^{-1}\exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\boldsymbol{\epsilon}^2_{\mathbf{Y}_i}\Big|_d\right\},\end{aligned}$$

$$\begin{aligned}\mathcal{N}(\mathbf{Z}_u^{(0)}|\boldsymbol{\mu}_{\mathbf{Z}_u}, \boldsymbol{\sigma}^2_{\mathbf{Z}_u}\mathbf{I}) &= \left[(2\pi)^{\frac{D}{2}}\prod_{d=1}^{D}\boldsymbol{\sigma}_{\mathbf{Z}_u}^{(0)}\Big|_d\right]^{-1}\exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{\mathbf{Z}_u^{(0)}-\boldsymbol{\mu}_{\mathbf{Z}_u}^{(0)}}{\boldsymbol{\sigma}_{\mathbf{Z}_u}^{(0)}}\Big|_d\right)^2\right\}\\ &= \left[(2\pi)^{\frac{D}{2}}\prod_{d=1}^{D}\boldsymbol{\sigma}_{\mathbf{Z}_u}^{(0)}\Big|_d\right]^{-1}\exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\boldsymbol{\epsilon}^2_{\mathbf{Z}_u}\Big|_d\right\},\end{aligned}$$

$$\mathcal{N}\left(\mathbf{Y}_i^{(T)}|\mathbf{0},\mathbf{I}\right) = (2\pi)^{-\frac{D}{2}}\exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\left(\mathbf{Y}_i^{(T)}\right)^2\Big|_{d_d}\right\},$$

$$\mathcal{N}\left(\mathbf{Z}_u^{(T)}|\mathbf{0},\mathbf{I}\right) = (2\pi)^{-\frac{D}{2}} \exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\left(\mathbf{Z}_u^{(T)}\right)^2\Big|_{d_d}\right\}, \tag{63}$$

where:

$$\boldsymbol{\epsilon}_{\mathbf{Y}_i} = \frac{\mathbf{Y}_i^{(0)} - \boldsymbol{\mu}_{\mathbf{Y}_i}^{(0)}}{\boldsymbol{\sigma}_{\mathbf{Y}_i}^{(0)}},$$
$$\boldsymbol{\epsilon}_{\mathbf{Z}_u} = \frac{\mathbf{Z}_u^{(0)} - \boldsymbol{\mu}_{\mathbf{Z}_u}^{(0)}}{\boldsymbol{\sigma}_{\mathbf{Z}_u}^{(0)}}.$$

Taking logarithm on both side of (63), we have:

$$\begin{aligned}
\log\mathcal{N}\left(\mathbf{Y}_i^{(0)}|\boldsymbol{\mu}_{\mathbf{Y}_i}^{(0)}, \left(\boldsymbol{\sigma}_{\mathbf{Y}_i}^{(0)}\right)^2\mathbf{I}\right) &= -\frac{D}{2}\log 2\pi - \sum_{d=1}^{D}\log\boldsymbol{\sigma}_{\mathbf{Y}_i}^{(0)}\Big|_d - \frac{1}{2}\sum_{d=1}^{D}\boldsymbol{\epsilon}_{\mathbf{Y}_i}^2\Big|_d,\\
\log\mathcal{N}\left(\mathbf{Z}_u^{(0)}|\boldsymbol{\mu}_{\mathbf{Z}_u}^{(0)}, \left(\boldsymbol{\sigma}_{\mathbf{Z}_u}^{(0)}\right)^2\mathbf{I}\right) &= -\frac{D}{2}\log 2\pi - \sum_{d=1}^{D}\log\boldsymbol{\sigma}_{\mathbf{Z}_u}^{(0)}\Big|_d - \frac{1}{2}\sum_{d=1}^{D}\boldsymbol{\epsilon}_{\mathbf{Z}_u}^2\Big|_d,\\
\log\mathcal{N}\left(\mathbf{Y}_i^{(T)}|\mathbf{0},\mathbf{I}\right) &= -\frac{D}{2}\log 2\pi - \frac{1}{2}\sum_{d=1}^{D}\left(\mathbf{Y}_i^{(T)}\right)^2\Big|_d,\\
\log\mathcal{N}\left(\mathbf{Z}_u^{(T)}|\mathbf{0},\mathbf{I}\right) &= -\frac{D}{2}\log 2\pi - \frac{1}{2}\sum_{d=1}^{D}\left(\mathbf{Z}_u^{(T)}\right)^2\Big|_d.
\end{aligned} \tag{64}$$

Substituting (64) and (17) into (62), we have the following:

$$\begin{aligned}
KL = \mathbb{E}_{q_\phi}&\left[\sum_{i\in\mathcal{W}}\left(-\frac{D}{2}\log 2\pi - \sum_{d=1}^{D}\log\boldsymbol{\sigma}_{\mathbf{Y}_i}^{(0)}\Big|_d - \frac{1}{2}\sum_{d=1}^{D}\boldsymbol{\epsilon}_{\mathbf{Y}_i}^2\Big|_d\right)\right.\\
&\left. - \sum_{t=1}^{T}\sum_{d=1}^{D}\boldsymbol{\sigma}_{\mathbf{Y}_i}^{(t)}\Big|_d - \left(-\frac{D}{2}\log 2\pi - \frac{1}{2}\sum_{d=1}^{D}\left(\mathbf{Y}_i^{(T)}\right)^2\Big|_d\right)\right]\\
+ \mathbb{E}_{q_\phi}&\left[\sum_{u\in\mathcal{U}}\left(-\frac{D}{2}\log 2\pi - \sum_{d=1}^{D}\log\boldsymbol{\sigma}_{\mathbf{Z}_u}^{(0)}\Big|_d - \frac{1}{2}\sum_{d=1}^{D}\boldsymbol{\epsilon}_{\mathbf{Z}_u}^2\Big|_d\right)\right.\\
&\left. - \sum_{t=1}^{T}\sum_{d=1}^{D}\boldsymbol{\sigma}_{\mathbf{Z}_u}^{(t)}\Big|_d - \left(-\frac{D}{2}\log 2\pi - \frac{1}{2}\sum_{d=1}^{D}\left(\mathbf{Z}_u^{(T)}\right)^2\Big|_d\right)\right]\\
= \mathbb{E}_{q_\phi}&\left[\sum_{i\in\mathcal{W}}\sum_{d=1}^{D}\left(-\frac{1}{2}\boldsymbol{\epsilon}_{\mathbf{Y}_i}^2 - \sum_{t=0}^{T}\log\boldsymbol{\sigma}_{\mathbf{Y}_i}^{(t)} + \frac{1}{2}\left(\mathbf{Y}_i^{(T)}\right)^2\right)\Big|_d\right]\\
+ \mathbb{E}_{q_\phi}&\left[\sum_{u\in\mathcal{U}}\sum_{d=1}^{D}\left(-\frac{1}{2}\boldsymbol{\epsilon}_{\mathbf{Z}_u}^2 - \sum_{t=0}^{T}\log\boldsymbol{\sigma}_{\mathbf{Z}_u}^{(t)} + \frac{1}{2}\left(\mathbf{Z}_u^{(T)}\right)^2\right)\Big|_d\right].
\end{aligned} \tag{65}$$