Temporal Context-Aware Representation Learning for Question Routing

Xuchao Zhang, Wei Cheng, Bo Zong, Yuncong Chen, Jianwu Xu, Ding Li, Haifeng Chen
NEC Laboratories America, Princeton, NJ, USA
xuczhang,weicheng,bzong,yuncong,jianwu,dingli,haifeng@nec-labs.com

ABSTRACT

Question routing (QR) aims at recommending newly posted questions to the potential answerers who are most likely to answer the questions. The existing approaches that learn users' expertise from their past question-answering activities usually suffer from challenges in two aspects: 1) multi-faceted expertise and 2) temporal dynamics in the answering behavior. This paper proposes a novel temporal context-aware model in multiple granularities of temporal dynamics that concurrently address the above challenges. Specifically, the temporal context-aware attention characterizes the answerer's multi-faceted expertise in terms of the questions' semantic and temporal information simultaneously. Moreover, the design of the multi-shift and multi-resolution module enables our model to handle temporal impact on different time granularities. Extensive experiments on six datasets from different domains demonstrate that the proposed model significantly outperforms competitive baseline models.

CCS CONCEPTS

• **Information systems → Social recommendation**; **Learning to rank**; *Temporal data*; *Personalization*; *Web log analysis.*

KEYWORDS

question routing; context-aware embedding; temporal dynamics

ACM Reference Format:
Xuchao Zhang, Wei Cheng, Bo Zong, Yuncong Chen, Jianwu Xu, Ding Li, Haifeng Chen. 2020. Temporal Context-Aware Representation Learning for Question Routing. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM'20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 INTRODUCTION

Community-based question answering (CQA) has become a popular web service where users can exchange information in the form of questions and answers. For instance, Quora[1], one of the most popular question answering sites, generates a question every 1.84 seconds and had accumulated up to 38 million questions as of January 2019. However, the rapid growth of CQA sites has led to a severe gap between the posted questions and the potential respondents. This causes question raisers to wait hours or even days for answers and makes respondents feel easily overwhelmed about selecting suitable questions to answer from the large number of open candidates. The question routing problem [14][38][17], an essential task to bridge the gap in CQA sites, aims to allocate the answerers more efficiently and find related questions for the answerers. Figure 1 shows a toy example of question routing in terms of two answerers and three questions. Answerers A1 and A2 answered *Tensorflow installation* related questions Q1 and Q3, respectively. Also, A2 is capable of answering the *NoSQL database* question. If we have a new question Q4 related to *Tensorflow*, both A1 and A2 who have equivalent expertise should be recommended. But if considering the temporal factor, A2 should be recommended since Q3 answered by A2 has more temporal closeness than A1. Moreover, since the questions in our task are described by natural language, the question routing task can be easily extended to other expert finding tasks described by text, such as bug triaging [28] and expert finding in social networks [1].

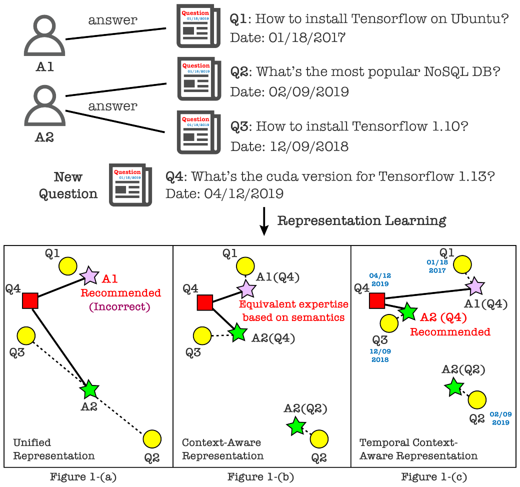


Figure 1: A toy example of question routing task with two answerers and three questions in two expertise domains.

Existing question routing approaches typically focus on modeling user expertise into a unified embedding vector [17][20][34] by the semantics of the questions they answered. However, these approaches suffer from the following two key challenges: **(i) Multi-faceted expertise**. Most of the users on CQA sites have multi-faceted expertise and are capable of answering questions in different domains. For instance, the answerer A2 in Figure 1 can take the

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

WSDM '20, February 3–7, 2020, Houston, TX, USA

ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/XXXXXX.XXXXXX

questions related to both *Tensorflow installation* and *NoSQL database*. Multi-faceted expertise cannot be explicitly modeled by the existing approaches that use a unified representation for answerers with multiple expertise. As shown in Figure 1-(a), the embedding of answerer A2 is trained by minimizing the sum of the distances of A2-Q2 and A2-Q3 when unified representation is used for answerer A2. Since answerer A1 only answered question 1, A1's embedding is close to Q1 and Q4. Thus, A1 is recommended in this case, which is contrary to the truth that both A1 and A2 have equivalent expertise on question 4 regarding *Tensorflow*-related questions in Figure 1-(b). **(ii) Temporal dynamics in the answering behavior.** The temporal dynamics of the answerers' interests are based on the observation that answerers may have prompt expertise or willingness to answer a question that they answered recently. As shown in Figure 1-(c), answerer A2, who answered the *Tensorflow* question recently, is more likely to answer the new *Tensorflow*-related questions again than answerer A1, who answered a similar question two years ago. Moreover, the granularity of the temporal dynamics is usually hard to define due to the characteristics of the answerers. For example, some answerers can keep answering questions for years, but others lose interest quickly.

In order to address the technical challenges above, this paper presents a novel temporal context-aware representation learning model for the question routing problem (TCQR). Specifically, the answerers are encoded into temporal context-aware representations in the context of the semantic and temporal information of the questions. Then the expertise of the answerers on certain questions will be measured as the coherence between the context-aware representations of the answerers and the encodings of the questions. Moreover, the *multi-shift* and *multi-resolution* extensions are proposed to model the temporal dynamics of the answering behaviors in different levels of time granularities. In addition, new triplet loss functions based on the answerers' ranking order and temporal dynamics are proposed to learn the users' answering behavior. The main contributions of this paper are summarized as follows: (i) **Design a temporal context-aware attention model to learn the answerer representation.** Specifically, instead of representing the answerer with a unified embedding, our model learns the answerer representation in the context of a question's semantic and temporal information, which helps to model multi-faceted expertise. (ii) **Propose a novel approach to model temporal dynamics by multi-shift and multi-resolution settings.** In particular, the multi-shift module is designed to model the temporal impact on the neighboring time periods and the multi-resolution setting is to control the temporal impact on both fine and coarse granularities. (iii) **Conduct extensive experiments to evaluate the performance of TCQR model.** Our proposed method was evaluated using six different datasets in different domains from hundreds of answerer candidates. The results demonstrate that the proposed approach significantly outperforms the state-of-the-arts alongside multiple metrics.

The rest of this paper is organized as follows. The related work is discussed in Section 2. Section 3 introduces the problem formulation. The proposed temporal context-aware question routing model is presented in Section 4. Experiments on real-world datasets are presented in Section 5, and the paper concludes with a summary of the research in Section 6.

2 RELATED WORK

In this section, we briefly review prior work on question routing, recommender system and context-aware embedding.

2.1 Question Routing

The majority of question routing work [7][31][32][20] falls into three categories: **(i) Feature engineering based methods**. The feature engineering approaches [38][14] extract features from users and questions and feed these features to the models, such as linear regression [2] and ranking models [14][26], to make recommendations of answerers for questions. However, these approaches rely on hand-crafted feature extraction, which is time-consuming and requires expert knowledge. **(ii) Matrix factorization based methods**. Matrix factorization models [35][5] decompose feature matrices based on the assumption that users may answer similar questions. Zhao et al. [35] considered the expert finding problem from the viewpoint of missing value estimation and designed a graph-regularized matrix completion algorithm to infer the user model. However, these matrix factorization based methods always suffer from the limitation of bag-of-word features and have difficulty in preserving the semantics of questions [17]. **(iii) Network embedding based methods**. Network embedding models construct a network based on the relation of questions and answerers and learn the representation of them in low-dimensional vectors, which preserve the structural context of nodes. Zhao et al. [34] exploited both users' relative quality rank to the questions and their social relations. Li et al. [17] learned the representations of a question, question raiser, and answerer by using a heterogeneous information network embedding algorithm. However, all the representation learning methods based on network embedding usually use a unified representation which cannot model multi-faceted expertise.

2.2 Recommender Systems

Traditional recommending approaches [25], such as collaborative filtering [24] and low-rank factorization [27], usually aim at recommending existing items to given users by learning the interactions between items and users, which makes it difficult to deal with new items such as newly raised questions that swiftly expire [37]. For instance, Guo et al. [36] proposed a deep factorization machine to combine the power of factorization machines for recommendation and deep learning for feature learning. Zhang et al. [33] utilized self-attention to infer item-item relationships from users' historical interactions. Some studies in recommender systems focus on recommending new items. Okura et al. [21] proposed an embedding-based method to use distributed representations for new item recommendation. However, the aforementioned recommendation method cannot be directly applied to the question routing problem due to the problem's specific characteristics of multi-faceted expertise and temporal dynamics of answering behavior.

2.3 Context-Aware Embedding

Recently, context-aware embedding has been utilized in many areas such as sentiment analysis[18], network analysis [29], recommending systems [12], and multimedia retrieval [8]. For instance, Liang et al. [18] proposed a context-aware embedding approach for the targeted aspect-based sentiment analysis problem by utilizing a sparse coefficient vector to adjust the context-aware embedding of target and aspect. Tu et al. [29] learned context-aware network embedding for a relation model in network analysis to represent the diverse roles of nodes. He et al. [15] proposed a context-aware recommendation model by capturing the contextual information of documents. However, most of the approaches consider a single modality of context, which cannot be applied to our multi-modal contexts for both question semantics and temporal information. Moreover, the hierarchical context-aware attention extension in multi-shift and multi-resolution enables our approach to model the temporal impact on neighboring periods in fine and coarse granularities.

3 PROBLEM FORMULATION

In this section, we first present the required notation and formulate the problem of question routing in community-based question answering (CQA) sites.

A CQA dataset that conserves all the question-answer sessions can be represented by the following sets: (i) **Question set** $Q = \{q_1, q_2, \ldots, q_n\}$, where n denotes the number of questions. Each question q_i can be represented as a tuple $q_i = (c_i, t_i)$, where c_i is the question content in natural language and t_i is the timestamp when the question was raised. (ii) **Answerer set** $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$, where m is the number of answerers. Each answerer a_i is represented by a low-dimensional embedding for the question routing task. (iii) **Question-Answer Session set** $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, where n is the total number of questions. Each question-answer session s_i includes all the answer information related to question q_i, and it can be represented as a tuple $s_i = (q_i, \Phi_i, \alpha_i)$, where the answerer set $\Phi_i \subseteq \mathcal{A}$ denotes all the answerers who answered the question q_i and $\alpha_i \in \Phi_i$ is the answerer who gave the unique accepted answer.

For example, if a question q_i raised on July 16, 2019 and answered by users a_1, a_4 and a_6, where a_4 is the answerer who provided the accepted answer, the question can then be represented as $q_i = (\langle \text{CONENT OF QUESTION} \rangle, 07/16/2019)$ and its question-answer session s_i is denoted by $s_i = (q_i, \{a_1, a_4, a_6\}, a_4)$.

To model the temporal dynamics of the answering behavior, we make the following definitions of time periods for further discussion in Section 4. First, we use different time resolutions to split the whole time period into units where the definition of time resolution is shown as follows:

Definition 3.1 (Time Resolution). Time resolution is the granularity to split a time period into multiple units. For instance, the time period from January 1, 2019 to July 1, 2019 can be split into 26 units by time resolution 7 days, where each time unit has 7 days except for the last time unit, which has 6 days.

Then we use the function $\delta(t)$ to represent the index of time unit belonging to time t. Following the previous example that splits the time period from 01/01/2019 to 07/01/2019, we have $\delta(t_1) = 1$ and $\delta(t_2) = 2$ when timestamp t_1 and t_2 are "01/02/2019" and "01/09/2019". Then the time shift between two timestamps can be defined as follows:

Definition 3.2 (Time Shift). Time shift $\Delta(t_i, t_j)$ between timestamp t_i and t_j is defined by $|\delta(t_i) - \delta(t_j)|$. For instance, if t_i and t_j are "01/02/2019" and "01/17/2019" respectively, then the time shift between them is 2.

Based on the definition of time resolution and shift, we can model the temporal impact on neighboring time units in fine and coarse granularities when applying t_i as the time of raising the question, which will be discussed in detail in Section 4.4.

Using the above notations, we define temporal context-aware question routing as the following: Given question set Q, answerer set $\mathcal{A}$ and a new question query $\hat{q} = (\hat{c}, \hat{t})$ where $\hat{c}$ and $\hat{t}$ are the content and raising timestamp of the new question, the question routing problem is to compute the ranking scores for each answerer $a \in \mathcal{A}$ and recommend the answerer with the highest ranking score as the predicted provider of the "accepted answer".

4 PROPOSED MODEL

In this section, we first demonstrate the overall architecture of the model. Then the details of temporal context-aware attention and temporal dynamics modeling via multi-shift and multi-resolution modules are provided.

4.1 Overall Framework

Our proposed Temporal Context-aware Question Routing model (TCQR) is a multi-layer deep neural networks integrating with temporal context-aware attention as well as multi-shift and multi-resolution temporal dynamics modules. The overall architecture is shown in Figure 2. The inputs of the model consist of both questions and answerers. Each answerer is represented by the embedding Matrix $U \in \mathcal{R}^{p \times d}$, where p is a hyper-parameter to control the scale of user expertise and d is the dimension for each user expertise. The answerer embedding is randomly initialized and is trainable by our proposed model. The question input contains both the question content c and question raising timestamp t. The content of the question is encoded by a pre-trained deep bidirectional Transformers model, BERT [6]. The encoding output by BERT is denoted by $Q \in \mathcal{R}^{l \times d}$, where l is the number of words in a question. By default, we choose the dimension of word that is the same value as the dimension of the answerer's embedding and fix the embedding of question content untrainable for fine-tuning. The question raising time is encoded into a unique representation vector $\boldsymbol{t} \in \mathcal{R}^d$ by the time encoding module, where the representation is also used to reflect the ordered sequence of the timeline. All the details of time encoding are explained in Section 4.2.

The content encoding and time encoding of question and answerer embedding will be used as the inputs of the Temporal Context-Aware (TCA) attention module, which aims to generate the answerer embedding $\boldsymbol{z} \in \mathcal{R}^d$ in the context of the question and its corresponding raising time. The details of the TCA attention module will be explained in Section 4.3. Then we employ the multi-shift and multi-resolution extension on the temporal context-aware

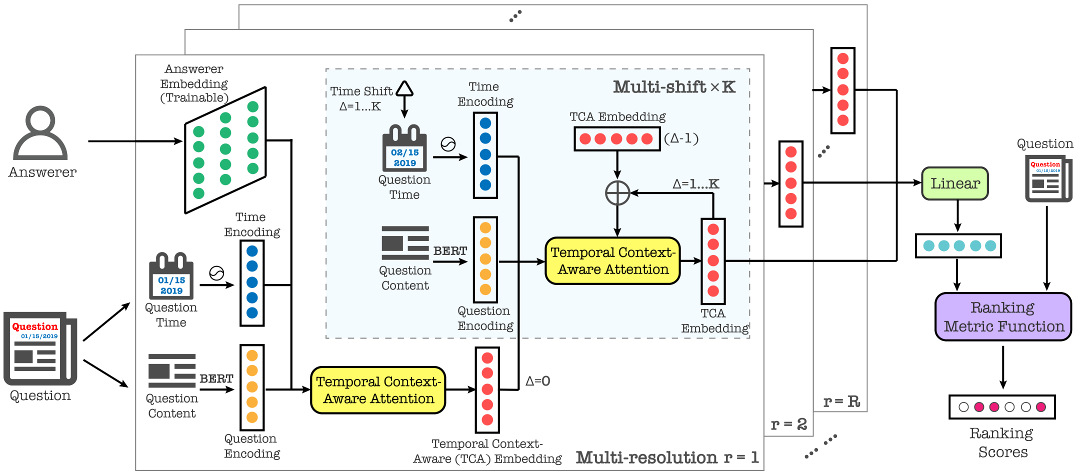


Figure 2: Overall Architecture of Proposed Model.

embedding to model the temporal dynamics on neighboring time periods via different granularities. The details of the multi-shift and multi-resolution extension are described in Section 4.4. After that, we use the ranking metric function σ to quantify the quality of answerer a for answering question q, which is defined as follows:

$$\sigma(Q, \boldsymbol{t}, \boldsymbol{z}) = \left(\mathrm{Avg}_{\mathrm{pool}}(Q) \oplus \boldsymbol{t}\right) \cdot \boldsymbol{z}^T, \tag{1}$$

where Q and $\boldsymbol{t}$ are the encoding of the question content and question raising time, respectively. The temporal context-aware embedding of the answerer is denoted by $\boldsymbol{z}$, and $\oplus$ is the operator to combine the question content and time. By default, we use the "add" operator since it has the similar performance as concatenation operator but takes less computational memory space. Then the coherence score will be utilized in the training process, which is described in Section 4.5.

4.2 Question Time Encoding

To encode the question raising timestamp into a low-dimensional representation $\boldsymbol{t} \in \mathcal{R}^d$, we employ a traditional position encoding method [9], and the value of its k-th position in $\boldsymbol{t}$ is defined as follows:

$$\boldsymbol{t}(k, j) = \begin{cases} \sin(k/10000^{j/d}) & \text{if } j = 2i - 1,\ i \in \mathcal{Z}^+ \\ \cos(k/10000^{j/d}) & \text{if } j = 2i,\ i \in \mathcal{Z}^+, \end{cases} \tag{2}$$

where d is the dimension of the time encoding and $\mathcal{Z}^+$ represents the positive integers starting from one. An example of time encodings from September 2008 to April 2019 with the time unit of 30 days is presented in Figure 3. Each row represents the time encoding for each time unit with 768 dimensions. The time encoding method satisfies the following two properties, which is necessary for our temporal dynamics modeling: **(i) Uniqueness**. The value of time encoding is unique when it represents different timestamps. **(ii) Sequential ordering**. The L2 norm distance between time encodings can be used to reflect the temporal distance. For example, when $\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3$ represent the dates 04/01/2019, 05/01/2019, and 06/01/2019, respectively, the following property is satisfied: $\|\boldsymbol{t}_1 - \boldsymbol{t}_2\|_2 \leq \|\boldsymbol{t}_1 - \boldsymbol{t}_3\|_2$.

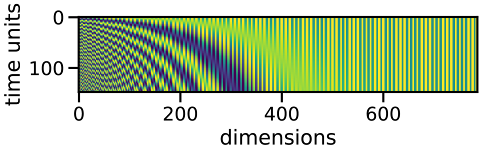


Figure 3: Visualization of time encoding from 09/2008 to 04/2019 by monthly granularity .

4.3 Temporal Context-Aware Attention

We will first explain the motivation of the temporal context-aware attention component and then give a detailed description of the component. First of all, most of the existing approaches assume the embeddings of two answerers are similar if both of them answered similar questions. However, this assumption cannot always be true when answerers have multi-faceted expertise. For example, if two answerers a_1 and a_2 are capable of answering questions in one area, according to the assumption, their representation $\boldsymbol{u}$ and $\boldsymbol{v}$ should be similar: $\boldsymbol{u} \approx \boldsymbol{v}$. However, if a_1 can also answer questions in a different area but a_2 cannot, their representation should be considered as different. Hence, in our model, we assume the embedding of an answerer is not unified but varied for different questions. Specifically, the embeddings of the two answerers are similar under the context of question q, $\boldsymbol{u}^{(q)} \approx \boldsymbol{v}^{(q)}$ when both of them answered the question, where $\boldsymbol{u}^{(q)}$ and $\boldsymbol{v}^{(q)}$ represent the two answerers' embeddings in the context of question q.

Following the multi-headed self-attention framework [30], we design our multi-headed temporal context-aware attention module, which is shown in Figure 4. Specifically, we combine the time encoding and question content encoding as the context to learn the attention between question and answerer. After that, we apply the attention to the answerer embedding for generating the temporal context-aware embedding $\boldsymbol{z}_k$ in terms of the k-th time shift, which

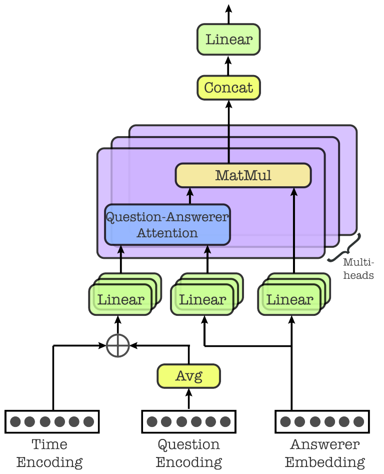


Figure 4: Temporal Context-Aware Attention.

is represented in Equation (3) as follows.

$$z_{k+1} = \text{softmax}\left(\frac{\left(\text{Avg}_{\text{pool}}\left(Q\right) \oplus \boldsymbol{t}_k\right) W_Q \left(z_k W_1\right)^T}{\sqrt{d}}\right) z_k W_2, \tag{3}$$

where $W_Q, W_1, W_2 \in \mathcal{R}^{d \times d}$ are the weights for the linear components. The embedding of the question content is denoted by $Q \in \mathcal{R}^{l \times d}$ and $\boldsymbol{t}_k \in \mathcal{R}^d$ represents the encoding of the timestamp in the k-th time shift. z_k denotes the embedding of the answerer that has separate representations in different values of time shift k. In particular, when $k = 0$, we have z_0 equals to the initial answerer embedding $U \in \mathcal{R}^{p \times d}$ without context information. Then the attention learned is a $d \times k$ matrix to show the relation between the question's semantic features and the answerer's expertise. When $k \geq 1$, we have $z_k \in \mathcal{R}^d$ to represent the temporal context-aware embedding in terms of time shift k. Then the attention learned from question-answerer attention is a scalar to show the importance of each time shift.

4.4 Multi-Shift and Resolution Extension

In this section, we extend our temporal context-aware embedding into its multi-shift and multi-resolution settings. For the multi-shift extension, we use a different time encoding with a different time shift Δ from 1 to K, where K is set to the maximum number of time shifts modeled for temporal impact. For example, when Δ=1 and the question raising timestamp is t, the time encodings of the time units $\delta(t) - 1$ and $\delta(t) + 1$ will be combined as the input of TCA Attention module. In particular, the shifted time encoding are combined as the sum of time encoding of the backward time unit and forward time unit. Different from the TCA attention module used in the first layer, we use a residual block to enable a shortcut connection between different time-shifted embeddings. Specifically, we employ the context-aware embedding input of the k-th time-shift layer $z_k^{(in)}$ as the sum of both the input and output of the $(k-1)$ layer: $z_k^{(in)} \leftarrow z_{k-1}^{(in)} + z_{k-1}^{(out)}$.

For the multi-resolution extension, we can choose different time resolutions to split the time period into multi-grained units. For each resolution, the time encoding includes the temporal information in diverse levels of time granularities. After the multi-shift temporal context-aware embedding layers, we combine the context-aware embedding $z_k^{(r_i)}$ for each time resolution together, where the superscript r_i represents the size of the i-th time resolution. Then we employ a fully connected layer to project combined embeddings into a d-dimensional embedding vector.

4.5 Training Process

To train our model, we first apply a ranking triplet loss function to learn the relative rank between positive samples (users answered the question) and negative samples (users did not answer the question). Moreover, to distinguish the answerer who provided the accepted answer from the other answerers in the same question, we also add an additional ranking loss term between them. The ranking loss is shown in Equation (4).

$$\begin{aligned} \mathcal{L}_r = \sum_{q_i \in Q} \Bigg\{ & \sum_{\substack{z^+ \in \Phi_i \\ z^- \notin \Phi_i}} \max\left(\sigma\left(Q_i, \boldsymbol{t}_i, z^+\right) - \sigma\left(Q_i, \boldsymbol{t}_i, z^-\right) + \alpha_p, 0\right) \\ & + \sum_{z^+ \in \Phi_i} \max\left(\sigma(Q_i, \boldsymbol{t}_i, z^*) - \sigma(Q_i, \boldsymbol{t}_i, z^+) + \alpha_c, 0\right) \Bigg\}, \end{aligned} \tag{4}$$

where Φ_i denotes the users who answered the question $q_i = (Q_i, \boldsymbol{t}_i) \in \boldsymbol{Q}$. The variables z^+, z^- and z^* represent the embedding of the positive answerers, negative answerers, and answerer who provided the accepted answer, respectively. We employ a margin value α_p to control the distance between positive and negative answerers and use a margin value α_c for the distance between positive answerers and the user who give the accepted answer.

Moreover, to learn the observation that more recent answering behaviors have higher impact on the recommendation of answerers, we propose a new temporal loss function between the neighboring time shifts in Equation (5).

$$\mathcal{L}_s = \sum_{q_i \in Q} \left\{ \sum_{z^+ \in \Phi_i} \sum_{k=1}^{K} \max\left(\sigma\left(Q_i, \boldsymbol{t}_i, z_{k-1}^+\right) - \sigma\left(Q_i, \boldsymbol{t}_i, z_k^+\right) + \alpha_s, 0\right) \right\}, \tag{5}$$

where k is the index of time shift and K is the total number of time shifts. z_k^+ represents the temporal context-aware embedding of answerers after the k-th time shift. We set the margin parameter α_s to one. Then we combine both the ranking loss and time shift loss together to generate the total loss $\mathcal{L}$ as follows: $\mathcal{L} = \mathcal{L}_r + \lambda \mathcal{L}_s$, where λ is a parameter to balance the two loss functions and is set to 0.5 by default.

4.6 Implementation Details

The model, TCQR, described in this section is implemented using the Pytorch[2] 1.0 framework and trained on four 12GB-memory RTX 2080ti GPUs in a 64-bit machine with 32 Intel Xeon@2.10GHz CPUs and 192GB memory. The question content is initialized by pre-trained deep bidirectional transformer (BERT) [6] using the default dimensionality of 768 and the maximum length of question content is set to 300 tokens. The embedding of each answerer is

[2] https://pytorch.org/

Table 1: Statistics of the Datasets

Dataset	Questions	Answerers	Time Range
ai	1130	163	2016.08-2019.06
bioinformatics	915	107	2017.05-2019.05
3dprinting	963	120	2016.01-2019.05
ebooks	368	74	2013.12-2019.05
history	4807	473	2011.05-2019.05
philosophy	4295	658	2011.04-2019.06

randomly initialized by a $l \times d$ matrix, where we choose the expertise size l to 20 and the dimension d of each expertise to 768. For the setting of temporal dynamics modeling, we set the number of time shifts to 3 and choose the time resolutions as 1, 2, and 3 with the minimum time unit as 30 days. For the training process, we set the number of training epochs to 50, the batch size to 16 and choose the Adam optimizer [16] to train our model with the learning rate 1e-5.

5 EXPERIMENT

In this section, the performance of the proposed model, TCQR, is evaluated using six real-world datasets. First, the experimental setup is introduced. Then the performance of the proposed model in terms of four different metrics is evaluated against several existing methods. In addition, we analyze the individual components with an ablation study and demonstrate the impact of temporal dynamics via a case study. Last, parameter sensitivity analyses of time shift and resolution are provided.

5.1 Experimental Settings

5.1.1 Datasets. We employed six real-world CQA datasets from StackExchangeto evaluate the performance of our model. All the datasets are publicly available[3]. The details of the datasets are presented in Table 1, including the number of questions, number of answerers, and their start and end dates. Each dataset contains all questions and their corresponding answer records, including the lists of answerers and the respondents who provided the accepted answers. Also, both question content and question raising timestamp are included in the datasets. We reserved the latest 20% of the data in the order of question raising time for the testing set and randomly split the remainder between 70% for training and 10% for validation. Both the answerers and the accepted answer for each question will be used as ground truth for evaluating the performance of our question routing model. Following the settings in Li et al. [17], we filtered the users who provided less than five answers out of the training set to avoid the cold start problem.

5.1.2 Metrics. We use common ranking evaluation metrics from the literature to evaluate our models. We consider all the users who answered question q as the candidate answerer set for question q and the user providing the accepted answer as the ground truth of recommendation. The metrics we used include: (i) **Mean Reciprocal Rank (MRR)** [4]: the average multiplicative inverse of the rank of the correct answer, represented mathematically as MRR $= \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i}$, where N is the number of samples and rank_i is the rank assigned to the ground truth answerer, who provided the accepted answer, by a model. (ii) **Precision@K**: The proportion of predicted instances where the ground truth answerer appears in the ranked top-K result. For example, P@3 or "Precision at 3" corresponds to the percentage of cases where the true answerer appears in the top 3 ranked results. We vary the value of K from 1 and 3 in our experiments. (iii) **Normalized Discounted Cumulative Gain (NDCG)** [13]: the normalized gain of each answerer based on its ranking position in the results. Specifically, the DCG at the p-th position is defined by $\text{DCG}_p = \frac{1}{p} \sum_{i=1}^{p} \frac{\text{rel}_i}{\log_2(i+1)}$, where *rel* represents the relevance score of all the answerers corresponding to the question. We set the *rel* scores of the users who answered the question to one and those of the users who did not answer the question to zero. The NDCG is defined by: $\text{NDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p}$, where IDCG is the ideal discounted cumulative gain.

5.1.3 Competing Methods. The experiment utilizes five comparison methods: (i) **Frequency**: The Frequency method ranks the users by the number of answers provided by each user. Then we use the frequency as the probability to randomly generate the ranking list of all the answerers. The experiment results are averaged by 50 trials. (ii) **Vote**: This method ranks a user by the number of positive votes minus the number of negative votes, averaged over all the answers the user has attempted. Similar to the Frequency method, we use the vote score to generate a ranking list and average the results by 50 trials. (iii) **Doc2Vec**: This method recommends the answerers who have previously answered the questions most relevant to the new given question. The state-of-the-art document embedding model, InferSent [3], is applied to compute the similarity between questions. We use the pre-trained 300-dimensional word vectors from fastText[19], which is trained on Common Crawl containing 600B tokens. (iv) **DeepFM** [11]: The DeepFM approach applies a deep-learning-based factorization machine to learn both low and high-order feature interaction. We applied Term Frequency and Inverse Document Frequency (TF-IDF) [23] as the feature input for the DeepFM model by filtering the stop words and choosing the top 2,000 features with highest TF-IDF scores. (v) **NeRank** [17]: The model jointly learns the representation of question content, raiser and answerers via a heterogeneous information network embedding algorithm. We applied default experiment settings with a meta-path length of 13, node coverage of 20, and window size of Skip-gram model of 4.

5.2 Performance of Question Routing

Table 2 summarizes the results of our temporal context-aware question routing model, TCQR, in six real-world datasets. From the results, we can conclude that our model significantly outperforms all the baselines in every dataset on all metrics. Specifically, our model achieves a 12.9% performance gain on average for MRR compared to the best baseline method, NeRank, which demonstrates that our model can recommend the users provided the accepted answer in a higher-ranking position. Since we only have one true answerer who provided the accepted answer for each question, it is extremely challenging to rank the user in the first position from hundreds of candidates. But our model still can achieve 31.8% precision of the P@1 metric on the *3dprinting* dataset, which is

[3] https://archive.org/details/stackexchange

Table 2: Performance on Question Routing

	ai				bioinformatics				3dprinting			
	MRR	P@1	P@3	NDCG	MRR	P@1	P@3	NDCG	MRR	P@1	P@3	NDCG
Frequency	0.045	0.009	0.028	0.208	0.048	0.010	0.037	0.199	0.078	0.026	0.046	0.245
Score	0.045	0.010	0.028	0.203	0.039	0.008	0.020	0.203	0.047	0.009	0.036	0.197
Doc2Vec	0.086	0.000	0.000	0.271	0.146	0.100	0.100	0.321	0.060	0.000	0.000	0.261
DeepFM	0.032	0.014	0.018	0.184	0.039	0.005	0.011	0.215	0.041	0.015	0.026	0.214
NeRank	0.144	0.061	0.144	0.265	0.139	0.061	0.141	0.275	0.135	0.049	0.130	0.288
TCQR	**0.253**	**0.137**	**0.282**	**0.403**	**0.202**	**0.100**	**0.232**	**0.366**	**0.461**	**0.318**	**0.528**	**0.594**
	ebooks				**history**				**philosophy**			
	MRR	P@1	P@3	NDCG	MRR	P@1	P@3	NDCG	MRR	P@1	P@3	NDCG
Frequency	0.019	0.004	0.010	0.173	0.017	0.003	0.009	0.173	0.009	0.001	0.002	0.163
Vote	0.058	0.005	0.026	0.228	0.016	0.002	0.007	0.171	0.010	0.001	0.004	0.164
Doc2Vec	0.072	0.000	0.000	0.271	0.070	0.000	0.050	0.236	0.029	0.000	0.000	0.196
DeepFM	0.043	0.001	0.026	0.221	0.007	0.000	0.000	0.151	0.004	0.000	0.001	0.144
NeRank	0.254	0.160	0.266	0.346	0.124	0.044	0.130	0.275	0.119	0.055	0.116	0.269
TCQR	**0.359**	**0.190**	**0.456**	**0.519**	**0.206**	**0.093**	**0.224**	**0.383**	**0.248**	**0.135**	**0.289**	**0.401**

Table 3: Ablation Study

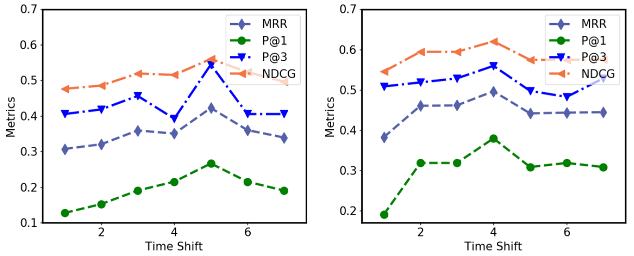
	ai		bios		3dprinting	
	MRR	NDCG	MRR	NDCG	MRR	NDCG
w/o BERT	**0.258**	0.393	0.181	0.347	0.365	0.533
w/o Temp	0.236	0.372	0.170	0.336	0.195	0.386
w/o S&R	0.189	0.353	0.144	0.319	0.447	0.593
TCQR	0.253	**0.403**	**0.202**	**0.366**	**0.461**	**0.594**
	ebooks		**history**		**philosophy**	
	MRR	NDCG	MRR	NDCG	MRR	NDCG
w/o BERT	0.339	0.489	0.164	0.342	0.221	0.384
w/o Temp	0.105	0.291	0.168	0.322	0.134	0.296
w/o S&R	0.315	0.469	0.158	0.345	0.145	0.327
TCQR	**0.359**	**0.519**	**0.206**	**0.383**	**0.248**	**0.401**

significantly better than the 4.9% precision of the best baseline method, NeRank. Our model similarly outperforms the best baseline in P@3 in all the datasets, obtaining an 18.3% improvement on average. The result of the NDCG metric also demonstrates our model can recommend not only the user who gave the accepted answer but all the users who answered the question. In particular, our model outperforms the best baseline method by 15.8% on average. From the results of the baseline methods, we can also conclude the following: 1) The two baseline methods based on the answerer's answering frequency and votes cannot perform competitively since they ignore the semantic information of the questions. 2) Doc2Vec recommends the users who answered the questions most similar to the new question. However, their performance is highly dependent on the temporal impact of answering behavior for the dataset. For instance, Doc2Vec performs better than the other baseline methods in the *bioinformatics* dataset since the time range of the dataset is relatively small. 3) DeepFM has a surprisingly low performance even compared to some intuitive baseline methods since it is hard to model the feature interaction on the textual features of questions. 4) NeRank has competitive experimental results in most of the datasets but still performs worse than our model since it applies a unified answerer representation and ignores the temporal impact of answering behavior.

5.3 Ablation Study

To verify the effectiveness of our modeling choices, we evaluate our model's performance in the absence of each of the following model components: (i) **w/o BERT**: Most of our competing methods apply traditional word embedding and recurrent neural network (RNN) [10] to model the text sequence. To demonstrate that the performance of our model is not simply due to using the pre-trained sequence encoder BERT, we replaced the BERT encoder with LSTM [10] on top of pre-trained 300-dimensional word vectors by Glove [22]. (ii) **w/o Temp**: To evaluate the impact of the temporal information, we eliminate the question's temporal information from the model and make the context-aware attention as follows: $z_{k+1} = \text{softmax}\left(\frac{\left(\text{Avg}_{\text{pool}}(Q)\right)W_Q\left(z_k W_1\right)^T}{\sqrt{d}}\right)z_k W_2$. (iii) **w/o Shift & Resolution**: To demonstrate the effectiveness of our multi-shift and multi-resolution extension, we set the size of both time shift and time resolution to one in this model variant.

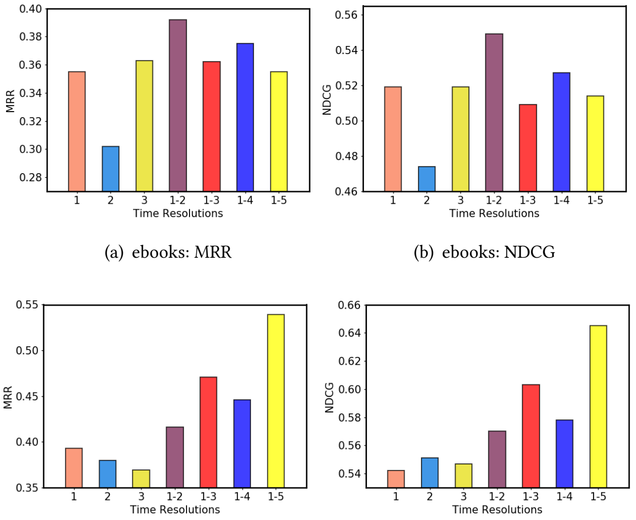
The results in Table 3 show that each component improves the overall performance on all the datasets across MRR and NDCG evaluation metrics. Specifically, the temporal information and multi-shift & resolution extensions improve 12% and 5.5% on average, respectively, over all the datasets in the MRR metric. This indicates that our modeling choices of both context-aware attention and temporal dynamics extension on answering behavior are particularly suitable to tackle the inherent challenges involved in the question routing task. Also, we found our model can still perform better than the best baseline method, NeRank, without using the BERT sequence encoder.



(a) **ebooks**

(b) **3dprinting**

Figure 5: Parameter Sensitivity Analysis on Time Shift.



(a) ebooks: MRR

(b) ebooks: NDCG

(c) 3dprinting: MRR

(d) 3dprinting: NDCG

Figure 6: Parameter Sensitivity Analysis on Time Resolution.

5.4 Temporal Effectiveness

To qualitatively evaluate the temporal impact on the question routing task, we show a case study on querying with the same content as an existing question but using multiple proceeding time shifts. Figure 7 shows the heat map of the coherence scores between the question and answerers, where each column represents the proceeding time shifts ranging from 0 to 14 and each row represents the six users who answered the question. The darker color represents a higher coherence score, while the lighter color indicates a lower score. From the results, we can see an obvious trend that the coherence scores become lower when the time shift becomes larger, which can be intuitively explained by the fact that users are less likely to answer a question that they answered a long time ago. Since we do not explicitly train our model in any temporal constraint of time shift larger than our default value of 3, the trend obtained on temporal dynamics is spontaneously learned by the data itself. Also, we found that the temporal pattern is different for each user. For instance, user 2 is willing to answer similar questions for a long time, but user 5 is reluctant to respond to a recently answered question.

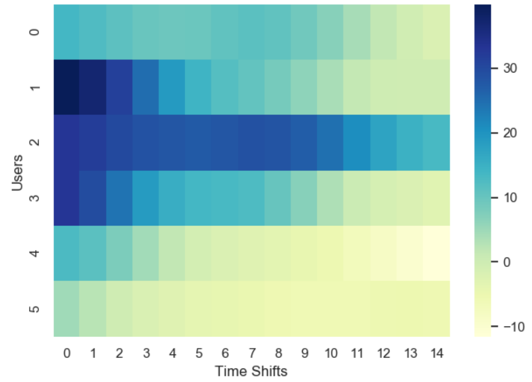


Figure 7: The case study of the temporal dynamics the answerers of one question in multiple time shifts.

5.5 Parameter Sensitivity

5.5.1 Time Shift. Figure 5 shows the parameter sensitivity analysis on the time shift with fixed time resolutions ranging from 1 to 3 with the default one-month time unit in the two datasets *ebooks* and *3dprinting*. We evaluated six different time shift settings ranging from 1 to 7. From the results, we found the performance is continuously improved when the size of the time shift is increased until a turning point. If the time shift is too large, the performance starts to degrade since it is easily overfitted to the training set. For instance, the performance of the dataset *ebooks* starts to degrade after the time shift is larger than 5.

5.5.2 Time Resolution. Figure 6 shows the parameter sensitivity analysis on the time resolutions in the *ebooks* and *3dprinting* datasets. We evaluated seven different time resolution settings, including three single-resolution settings (1, 2 and 3) and four multi-resolution settings (1-2, 1-3, 1-4 and 1-5, where the resolutions 1-5 represent the consecutive resolutions ranging from 1 to 5). Figures 6(a) and 6(b) show the results of the ebooks dataset in the metrics MRR and NDCG, respectively. From the results, we observe the multi-resolution settings have better overall performance than single resolution since they provide more flexibility to model the temporal dynamics in mutliple granularities. Although we observe that more resolutions usually have better performance, they are still impacted by the temporal characteristics of answering behavior for different datasets. For instance, the performance of resolutions 1-4 was worse than that of resolutions 1-3 in the *3dprinting* dataset, which is shown in Figures 6(c) and 6(d).

6 CONCLUSION

In this paper, we propose a temporal context-aware question routing model, TCQR, in community-based question answering (CQA) systems. Our model learns the answerers' representation in the context of both the semantic and temporal information to handle the multi-faceted expertise of answerers in CQA system. To model the temporal dynamics of answering behavior, we extend our temporal context-aware attention model into its multi-shift and multi-resolution extensions, which enable our model to learn the temporal impact on the neighboring time periods in multiple time granularities. Extensive experiments on several real-world datasets demonstrated the advantageous performance of the proposed model over the existing baselines.

REFERENCES

[1] Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Matteo Silvestri, and Giuliano Vesci. 2013. Choosing the right crowd: expert finding in social networks. In *Proceedings of the 16th International Conference on Extending Database Technology*. ACM, 637–648.

[2] S. Chang and A. Pal. 2013. Routing questions for collaborative answering in Community Question Answering. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*. 494–501. https://doi.org/10.1109/ASONAM.2013.6785750

[3] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 670–680. https://www.aclweb.org/anthology/D17-1070

[4] Nick Craswell. 2009. *Mean Reciprocal Rank*. Springer US, Boston, MA, 1703–1703. https://doi.org/10.1007/978-0-387-39940-9_488

[5] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Recurrent Coevolutionary Latent Feature Processes for Continuous-Time Recommendation. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. ACM, New York, NY, USA, 29–34. https://doi.org/10.1145/2988450.2988451

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://www.aclweb.org/anthology/N19-1423

[7] Min Fu, Min Zhu, Yabo Su, Qiuhui Zhu, and Mingzhao Li. 2016. Modeling Temporal Behavior to Identify Potential Experts in Question Answering Communities. In *Cooperative Design, Visualization, and Engineering*, Yuhua Luo (Ed.). Springer International Publishing, Cham, 51–58.

[8] Noa Garcia, Benjamin Renoust, and Yuta Nakashima. 2019. Context-Aware Embeddings for Automatic Art Analysis. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval (ICMR '19)*. ACM, New York, NY, USA, 25–33. https://doi.org/10.1145/3323873.3325028

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1243–1252.

[10] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2016. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems* 28, 10 (2016), 2222–2232.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[12] J. He, J. Qi, and K. Ramamohanarao. 2019. A Joint Context-Aware Embedding for Trip Recommendations. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 292–303. https://doi.org/10.1109/ICDE.2019.00034

[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[14] Zongcheng Ji and Bin Wang. 2013. Learning to rank for question routing in community question answering. In *Proceedings of the 22nd ACM international conference on Conference on information &#38; knowledge management (CIKM '13)*. ACM, New York, NY, USA, 2363–2368. https://doi.org/10.1145/2505515.2505670

[15] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 233–240. https://doi.org/10.1145/2959100.2959165

[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[17] Zeyu Li, Jyun-Yu Jiang, Yizhou Sun, and Wei Wang. 2019. Personalized Question Routing via Heterogeneous Network Embedding. In *AAAI 2019*.

[18] Bin Liang, Jiachen Du, Ruifeng Xu, Binyang Li, and Hejiao Huang. 2019. Context-aware Embedding for Targeted Aspect-based Sentiment Analysis. *arXiv preprint arXiv:1906.06945* (2019).

[19] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

[20] Sara Mumtaz, Carlos Rodriguez, and Boualem Benatallah. 2019. Expert2Vec: Experts Representation in Community Question Answering for Question Routing. In *Advanced Information Systems Engineering*, Paolo Giorgini and Barbara Weber (Eds.). Springer International Publishing, Cham, 213–229.

[21] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-based News Recommendation for Millions of Users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. ACM, New York, NY, USA, 1933–1942. https://doi.org/10.1145/3097983.3098108

[22] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[23] Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, Vol. 242. Piscataway, NJ, 133–142.

[24] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *Www* 1 (2001), 285–295.

[25] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 291–324.

[26] Mohsen Shahriari, Sathvik Parekodi, and Ralf Klamma. 2015. Community-aware Ranking Algorithms for Expert Identification in Question-answer Forums. In *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business (i-KNOW '15)*. ACM, New York, NY, USA, Article 8, 8 pages. https://doi.org/10.1145/2809563.2809592

[27] Yue Shi, Martha Larson, and Alan Hanjalic. 2010. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 269–272.

[28] Ahmed Tamrawi, Tung Thanh Nguyen, Jafar M Al-Kofahi, and Tien N Nguyen. 2011. Fuzzy set and cache-based approach for bug triaging. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*. ACM, 365–375.

[29] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. CANE: Context-Aware Network Embedding for Relation Modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 1722–1731. https://doi.org/10.18653/v1/P17-1158

[30] 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 neural information processing systems*. 5998–6008.

[31] Lin Wang, Bin Wu, Juan Yang, and Shuang Peng. 2016. Personalized Recommendation for New Questions in Community Question Answering. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '16)*. IEEE Press, Piscataway, NJ, USA, 901–908. http://dl.acm.org/citation.cfm?id=3192424.3192594

[32] Sha Yuan, Yu Zhang, Jie Tang, Wendy Hall, and Juan Bautista Cabotà. 2019. Expert finding in community question answering: a review. *Artificial Intelligence Review* (2019), 1–32.

[33] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. 2018. Dynamic Intention-Aware Recommendation with Self-Attention. *arXiv preprint arXiv:1808.06414* (2018).

[34] Zhou Zhao, Qifan Yang, Deng Cai, Xiaofei He, and Yueting Zhuang. 2016. Expert Finding for Community-based Question Answering via Ranking Metric Network Learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press, 3000–3006. http://dl.acm.org/citation.cfm?id=3060832.3061041

[35] Z. Zhao, L. Zhang, X. He, and W. Ng. 2015. Expert Finding for Question Answering via Graph Regularized Matrix Completion. *IEEE Transactions on Knowledge and Data Engineering* 27, 4 (April 2015), 993–1004. https://doi.org/10.1109/TKDE.2014.2356461

[36] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 425–434.

[37] Erheng Zhong, Nathan Liu, Yue Shi, and Suju Rajan. 2015. Building Discriminative User Profiles for Large-scale Content Recommendation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 2277–2286. https://doi.org/10.1145/2783258.2788610

[38] Tom Chao Zhou, Michael R. Lyu, and Irwin King. 2012. A Classification-based Approach to Question Routing in Community Question Answering. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12 Companion)*. ACM, New York, NY, USA, 783–790. https://doi.org/10.1145/2187980.2188201