Personalized Question Routing via Heterogeneous Network Embedding

Zeyu Li, Jyun-Yu Jiang, Yizhou Sun, Wei Wang
University of California, Los Angeles
Los Angeles, CA, 90095, USA
{zyli, jyunyu, yzsun, weiwang}@cs.ucla.edu

Abstract

Question Routing (QR) on Community-based Question Answering (CQA) websites aims at recommending answerers that have high probabilities of providing the "accepted answers" to new questions. The existing question routing algorithms simply predict the ranking of users based on query content. As a consequence, the question raiser information is ignored. On the other hand, they lack learnable scoring functions to explicitly compute ranking scores.

To tackle these challenges, we propose NeRank that (1) jointly learns representations of question content, question raiser, and question answerers by a heterogeneous information network embedding algorithm and a long short-term memory (LSTM) model. The embeddings of the three types of entities are unified in the same latent space, and (2) conducts question routing for personalized queries, i.e., queries with two entities `(question content, question raiser)`, by a convolutional scoring function taking the learned embeddings of all three types of entities as input. Using the scores, NeRank routes new questions to high-ranking answerers that are skillfulness in the question domain and have similar backgrounds to the question raiser.

Experimental results show that NeRank significantly outperforms competitive baseline question routing models that ignore the raiser information in three ranking metrics. In addition, NeRank is convergeable in several thousand iterations and insensitive to parameter changes, which prove its effectiveness, scalability, and robustness.

Introduction

Community-based question answering (CQA) such as Stack Overflow[1] is rapidly gaining popularity and becoming an important type of social media for sharing and spreading knowledge. Through CQA websites, users with questions are able to quickly locate answers provided by experts. A user can also create a new post if relevant and satisfactory QA records do not exist, and then wait for answers from the community. After several responses are gathered, the question raiser reviews the answers and selects one that he/she is the most satisfied with as the "accepted answer".

The answer collection can be unacceptably time-consuming due to the lack of an efficient way to find the "domain experts". As a result, a large number of questions remain poorly addressed.

One of the solutions to promote answer collection is to automatically identify users that tend to contribute high-quality answers and then send answer invitations to them. The answer collection is consequently accelerated since these users are able to immediately spot the questions of their expertise. Such task is also known as *question routing* and is previously addressed by feature engineering-based approaches (Zhou, Lyu, and King 2012; Ji and Wang 2013; Chang and Pal 2013). Features exploited include the statistics of users, the language modeling features of question content, and the relationships between users and questions. All of them focus on estimating users' authority level and identifying the skillful users for recommendation.

However, the feature engineering-based strategies have at least three limitations as follows. First, they are not personalized, i.e., they cannot customize recommendations for questions raised by users with diverse characteristics due to ignoring the background and preference of the question raisers. Second, they lack explicit definitions of scoring functions for queries with multiple entities and, therefore, have trouble computing scores for new question routing queries. Third, they model question content by language model or topic model features which are unable to capture the complex semantics of question content. Also, their representation power is undermined when handling questions with new topics that are unobserved or underrepresented in the training set.

In order to overcome the above limitations, we propose *NeRank*, which stands for Network embedding-augmented Ranking for question routing. NeRank assesses the "personalized authority", i.e., the authority of an answerer with respect to not only the question content but also the background of the question raiser, for question routing. In addition, the recommended answerers are also expected to have similarities with the question raiser in domains of interest to fulfill the "personalization" requirement so that their responses conform to the raiser's anticipation.

In particular, NeRank models CQA websites as heterogeneous information networks (HIN), namely *CQA networks*, and applies a metapath-based heterogeneous network embedding algorithm to CQA networks to learn representations for question raisers and question answerers. A long short-

[1] https://stackoverflow.com

term memory (LSTM) component is specifically utilized to learn question content representation. Using network embedding, the proximity information between the entities in a CQA network is preserved.

NeRank models the question routing task as a ranking problem and utilizes a convolutional neural network (CNN) to compute the ranking score of an answerer given a query `(question raiser, question content)`. Such ranking score measures the probability of the answerer providing the "accepted answer" to this question. Compared with previous frameworks (Zhou, Lyu, and King 2012; Zhao et al. 2015; Ji and Wang 2013), our ranking function explicitly computes the ranking scores taking advantage of rich non-linear information of the three entities.

We summarize our contributions as follows:

- We propose NeRank for personalized question routing on CQA websites. Compared with existing models, NeRank considers question raiser's profile in addition to question content. To the best of our knowledge, this is the first work on personalized Question Routing on CQA websites.
- We learn representations of entities by an HIN embedding method and compute ranking scores for answerers by an explicitly defined CNN scoring function given the learned representations as input. It is a novel attempt to apply an embedding-based method to question routing.
- We conduct extensive experiments on two real CQA datasets and evaluate the routing performance of NeRank via three ranking metrics. Our model outperforms the baselines in these metrics. Results also show that the HIN embedding algorithm and CNN scoring function improve the ranking performance.

Related Work

In this section, we introduce existing works on question routing, recommender systems, and network embedding.

Question Routing. Question routing is defined as predicting whether a user in CQA will share knowledge and answer a given question (Zhou, Lyu, and King 2012). The majority of previous works fall into two categories: feature engineering-based methods and matrix factorization-based methods.

Feature engineering algorithms (Zhou, Lyu, and King 2012; Ji and Wang 2013; Chang and Pal 2013) feed features extracted from users, questions, and their relations to models such as SVM (Hearst et al. 1998) and Linear Regression (Chang and Pal 2013) to rank the user authority to make recommendations. However, they require carefully crafted features and the performance relies heavily on feature selection.

Matrix factorization models decompose feature matrices based on the "low-rank" assumption to discover users' expertise on particular words and compute ranking scores by the inner product of the user and question feature vectors (Zhao et al. 2015). It suffers from the limitation of bag-of-word model which is unable to preserve sequential text semantics.

Recommender Systems. Recommender systems aim at recommending items to a given user. The development of deep learning models offers more possibilities to recommender system architectures. Gong and Zhang recommended hashtags for online microblogs using CNN. Okura et al. recommended news of interest using embeddings of text and users learned by autoencoder and recurrent neural networks such as long short-term memory (LSTM) and gated recurrent unit (GRU). The aforementioned methods cannot deal with new items, i.e. new questions in our problem, and thus are unable to be applied to question routing.

Network Embedding. Network embedding models learn low-dimensional representations for nodes in a network that preserve the structural context of nodes (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016; Tang et al. 2015; Shi et al. 2018; Dong, Chawla, and Swami 2017; Chen and Sun 2017). For HIN, the diversified node and edge types bring forth additional semantic information of networks which motivates the metapath-based network embedding algorithms (Sun et al. 2011). Chen and Sun proposed a task-specific and path-augmented model that jointly optimized the network-general and task-specific objectives. Metapaths were specifically selected for the task. *metapath2vec* and *metapath2vec++* (Dong, Chawla, and Swami 2017) combined metapaths with word2vec model for heterogeneous embedding learning.

Preliminaries and Problem Statement

A CQA network is built upon a static archive of a CQA website conserving all question-answer sessions accumulated over time. We create question Raiser set $R = \{r_1, r_2, \ldots, r_m\}$ and Answerer set $A = \{a_1, a_2, \ldots, a_k\}$ where m is the number of users who have asked questions, i.e. question raisers, and k is the number of users who have answered questions, i.e. question answerers, in this CQA website. Note that we only model users that have asking or answering records in the dataset. Hence, each user of the CQA website may have one or two embeddings associated with the role(s) they played. We create Question set $Q = \{q_1, q_2, \ldots, q_l\}$ where l denotes the number of questions. There exist two relations among these entities, namely "raises a question" between entities in R and Q and "answers a question" between entities in A and Q.

A *CQA network* is defined as a heterogeneous information network $G = (V, E, T, \phi)$, where $V = R \cup Q \cup A$ denotes the node set; E denotes the edge set; T denotes the set of three entity types involved (Zhao et al. 2017); $\phi : V \to T$ is a labeling function that maps an entity into its type $t \in T$. Each edge in a CQA network symbolizes an asking or answering record. Note that entities of R and A do not directly interact with each other and hence there is no connection between them. Figure 1 shows a toy example of a CQA network. Node r_2 is linked to q_2 and q_3, meaning that r_2 poses q_2 and q_3. q_3 is linked to a_3 and a_4 since a_3 and a_4 answer q_3. a_1 and a_2 have strong similarity since they both answer q_1 and q_2.

Using above notations, we define the personalized Question Routing as the following: Given a CQA network $G = (V, E, T, \phi)$ and a query `(question raiser, question content)` denoted by $\gamma = (r, q)$ where $r \in R$ is a question raiser and $q \in Q$ is a new question, com-

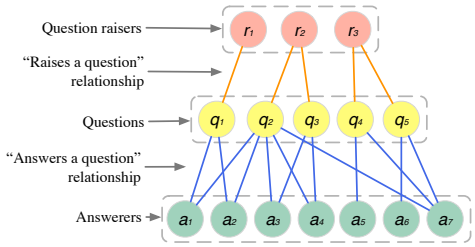


Figure 1: A Heterogeneous Network View of a CQA Website.

pute the ranking scores for answerers $a \in A$ and select the answerer with the highest ranking score as the predicted provider of the "accepted answer".

NeRank Framework

In this section, we demonstrate the technical details of NeRank. The list of notations is provided in Table 1 in advance for the convenience of later discussion.

Table 1: Frequently used notations.

Notation(s)	Definition
G	The CQA network.
E, V, T	The edge set, node set, and type set of G.
d	The dimension of entity embeddings.
v_e, u_e	The d-dimensional embedding of entity e.
n, c	The center and context entity (Mikolov et al. 2013b).
$\mathcal{P}$	A metapath.
$w_{\mathcal{P}}$	A walk generated according to $\mathcal{P}$.
τ_j	The entity type of the j-th element of $\mathcal{P}$.
$e_t^{(i)}$	The i-th entity of $w_{\mathcal{P}}$ with type τ_t.
$\phi(e)$	The entity type of the entity e.
D	The corpus of all (n, c) pairs, positive samples.
$\mathbf{\Theta}$	The parameter set of NeRank.

Overview

We formalize the personalized question routing problem as a ranking task in NeRank which ranks the probabilities of potential answerers contributing the "accepted answers" using the embeddings of entites. Specifically, it has two steps: modeling entity-wise similarity and computing the ranking scores of answerers given `(question raiser, question content)` queries. The trained representations for the three types of entities are expected to preserve both the entity-wise *proximity* information and the question-raiser-specific *expertise* information.

In the following subsections, we explain how the NeRank pipeline (shown in Figure 2) acquires the embeddings with proximity and enterprise information and computes the rankings scores.

LSTM-equiped Metapath-based Embedding with Negative Sampling

To capture proximity information, we learn embeddings of heterogeneous entities using an LSTM-equipped metapath-based heterogeneous network embedding model. We first explain the metapath-based Skip-gram on HIN and then show the jointly optimized LSTM model for question content representation learning.

Metapaths for HIN Embedding. HIN owns various node types, which differs from homogeneous networks. Simply applying the original random walk-based Skip-gram to HINs results in biases towards certain types of nodes (Sun et al. 2011). To create a bias-free walk corpus for Skip-gram on HINs, Dong, Chawla, and Swami generate walks according to the patterns specified by *metapaths*. It has been proved that HIN embedding models benefit from metapaths in reducing biases (Dong et al. 2015; Sun and Han 2012; Sun et al. 2013).

A metapath $\mathcal{P}$ is a sequence of objects linked by relations in form of $\tau_1 \xrightarrow{\pi_1} \tau_2 \xrightarrow{\pi_2} \dots \tau_t \xrightarrow{\pi_t} \tau_{t+1} \dots \xrightarrow{\pi_{l-1}} \tau_l$. $\pi = \pi_1 \circ \pi_2 \circ \dots \circ \pi_{l-1}$ denotes the composite relations between node types τ_1 and τ_l, $\tau_i \in T$. For example, metapath "$A \xrightarrow{\text{answers}} Q \xrightarrow{\text{raises}^{-1}} R \xrightarrow{\text{raises}} Q \xrightarrow{\text{answers}^{-1}} A$" means that two answerers each solves a question raised by the same person. τ^{-1} denotes the inverse relationship of τ, e.g. raises^{-1} represents "is raised by". Apparently, the metapath preserves semantic and structural correlations of entities in HIN which will be encoded in the representations by the Skip-gram model. We will omit the relations in metapath notations (e.g., "$AQRQA$") in the following discussion since the relation type between each given pair of entities is unique.

Metapath $\mathcal{P}$ guides the walk generation as follows. Randomly select an entity e_1 of type τ_1 as the initial entity, and then cycle from nodes of type τ_2 to τ_l until $w_{\mathcal{P}}$ grows to the desired length L. An example walk in Figure 1 is $a_5 q_4 r_3 q_5 a_7 q_2 r_2 q_3 a_4$ given $\mathcal{P}$ ="$AQRQA$" and $L = 9$. The transition from τ_t-type entity $e_t^{(i)}$ to entity $e^{(i+1)}$ is governed by the transition probability $p(e^{(i+1)}|e_t^{(i)}, \mathcal{P})$:

$$p(e^{(i+1)}|e_t^{(i)}, \mathcal{P}) = \begin{cases} \frac{1}{N} & (e^{(i+1)}, e_t^{(i)}) \in E, \phi(e^{(i+1)}) = \tau_{t+1} \\ 0 & \text{otherwise.} \end{cases},$$

where N denotes the count of τ_{t+1}-type neighbors of $e_t^{(i)}$.

Skip-gram with Negative Sampling on HINs. Negative sampling is an approximation strategy to relieve the expensive computational cost of softmax function (Mikolov et al. 2013a). Skip-gram with negative sampling maximizes the likelihood of D, the positive sample set generated from the metapath walk corpus by the sampling method in (Mikolov et al. 2013a), and minimizes the likelihood of the negative samples $D' = \{(n, c)|n, c \in V \wedge (n, c) \notin D\}$. The overall likelihood $\mathcal{L}(D, D'|\mathbf{\Theta})$ to maximize is:

$$\mathcal{L}(D, D'|\mathbf{\Theta}) = \sum_{D} \log(\sigma(v_n \cdot u_c)) + \sum_{D'} \log(-\sigma(v_n \cdot u_c)). \quad (1)$$

In Equation (1), v_n and u_c are representations of center entity n and context entity c. They are parts of the model parameter $\mathbf{\Theta}$. $\sigma(\cdot)$ is the sigmoid function. Equation (1) is consistent with word2vec that each entity has two versions of embeddings. We select the "center entity" version embedding as the input of the CNN recommemder model.

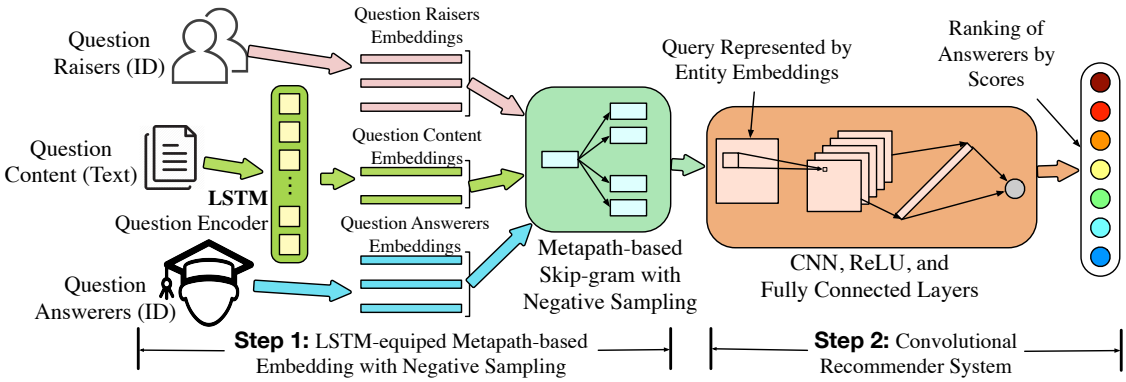


Figure 2: The pipeline of NeRank. There are two steps in the training procedure. Step 1 learns the entity embeddings using an LSTM-equipped metapath-based network embedding algorithm. Step 2 computes the ranking scores using a convolutional scoring function and finally outputs a ranked list of answerers.

Here we emphasize the necessity of the HIN embedding component without which the pure CNN scoring function has limited capability of analyzing user-user correlations and similarity.

LSTM for Question Representation. Different from the question raisers and answerers whose embeddings are parts of the parameter $\mathbf{\Theta}$, the embeddings of question content, v_q, are not in $\mathbf{\Theta}$ but directly obtained from question text through an LSTM model. v_q is then sent to Equation (1) together with the corresponding v_r and v_a for training and ranking score calculations.

LSTM is powerful in learning sequential features such as the semantics of text and has been applied to a variety of tasks such as text classification (Zhou et al. 2015) and machine translation (Bahdanau, Cho, and Bengio 2014). We skip the mathematical details of LSTM since they have been frequently discussed in previous literature.

The generation of v_q of question q with L_q words is as follows. The input is the word embedding matrix of question $\boldsymbol{x} \in \mathbb{R}^{L_q \times k}$ composed of k-dimensional word vectors. The LSTM cell at the t-th time step receives the t-th word embedding vector $x_t \in \mathbb{R}^k$ in $\boldsymbol{x}$ as well as the hidden state h_{t-1} from the previous time step. The output at time t is the hidden state $h_t \in \mathbb{R}^d$ which contains the accumulated semantic information from x_0 to x_t. Therefore, we use the hidden state output of the last time unit, h_{L_q}, as the text semantic representation v_q for the textual content of q.

It is worth mentioning that given the trained $\mathbf{\Theta}$ and the word sequence of a new question q_{new}, the representation learning of q_{new} is independent of the training data and the CQA network structure. Therefore, the LSTM component tackles the challenge of cold start issues for new questions.

Convolutional Recommender System

In this section, we present a convolutional neural network ranking model F that comprehensively analyzes the correlations between the three entities and computes the ranking score. Considering the properties of a ranking score, we rationally assume the following two partial order constraints based on our intuition and observation made on the dataset: (1) The best answerer has the highest score among all answerers to the query $\gamma = (r, q)$; (2) Answerers who answered q have higher scores than those who did not.

Using entity representation v_r, v_q, and v_a, we translate the above constraints and formalize the scoring function $F(v_r, v_q, v_a)$ as follows:

$$\begin{aligned} &\forall a^*, a \in A_\gamma, \forall a_n \in A \text{ and } a_n \notin A_\gamma, \\ &F(v_r, v_q, v_{a^*}) \geq F(v_r, v_q, v_a), \\ &F(v_r, v_q, v_a) \geq F(v_r, v_q, v_{a_n}), \end{aligned} \tag{2}$$

where A_γ is the set of answerers of $\gamma = (r, q)$, $a^* \in A_\gamma$ is the accepted answerer, and a_n is an answerer that is not involved in γ.

The reason of building a CNN-based scoring function is as follows: CNN has a strong capability of extracting hidden correlations of entities represented by static feature maps such as images (He et al. 2016) and text (Kim 2014). Compared with some straightforward scoring functions such as the dot-product, CNN is more powerful in preserving sophisticated correlations in the embedding matrices. Therefore, we design F as a CNN since the ranking score produced by $F(v_r, v_q, v_a)$ can be considered as a hidden feature of the combination of r, q, and a.

The computation of ranking scores is depicted in Figure 3. Given a query $\gamma = (r, q)$ and an answerer a to compute ranking score for, we stack their embeddings to construct the feature map $\boldsymbol{M}$ as $\boldsymbol{M} = [v_r, v_q, v_a]$, $\boldsymbol{M} \in \mathbb{R}^{d \times 3}$. Three convolutional kernels $k_1 \in \mathbb{R}^{d \times 1}$, $k_2 \in \mathbb{R}^{d \times 2}$, and $k_3 \in \mathbb{R}^{d \times 3}$ are applied to the input matrices. The intermediate hidden features go through two fully-connected layers and ReLU layers before deriving the ranking score. k_1 extracts the hidden features within the vector of each entity. k_2 captures the correlation between (1) v_r and v_q and (2) v_q and v_a since they have direct interactions. k_3 extracts the overall correlations across the three entities. Therefore, the aggregation of k_1, k_2, and k_3 is able to comprehensively utilize the hidden features in $\boldsymbol{M}$ and measure the score of a given γ.

The inequalities in Equation (2) holds for all (1) the "accepted" triplets versus the corresponding "answered but un-

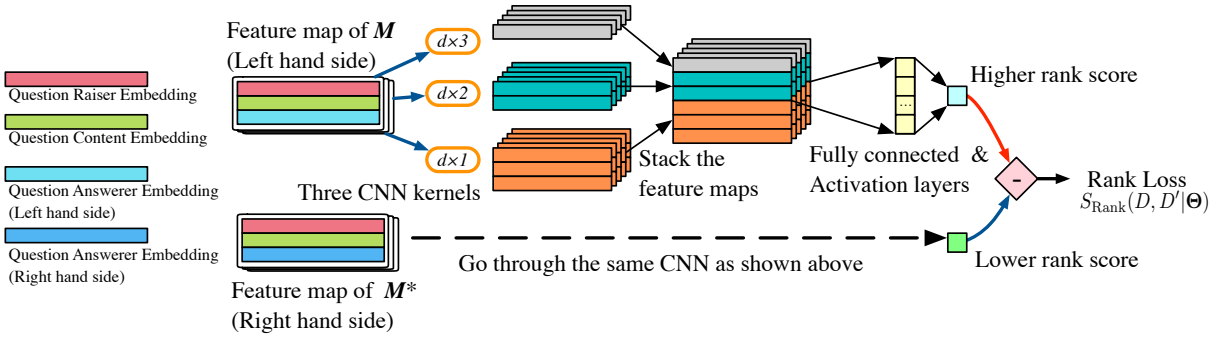


Figure 3: The CNN that learns the ranking scores and computes the ranking loss. From the left side, two matrices $\boldsymbol{M} \in \mathbb{R}^{d\times 3}$ (above) and $\boldsymbol{M^*} \in \mathbb{R}^{d\times 3}$ (below) are sent to a CNN followed by two fully connected layers and ReLU layers. The matrix $\boldsymbol{M}$ generates a lower ranking score and $\boldsymbol{M}^*$ generates a higher score. The difference between the scores is the ranking loss.

accepted" triplets and (2) the "answered" triplets versus the random "unanswered" triplets. Therefore, we model the ranking as the maximization of $S_{\text{Rank}}(D, D'|\boldsymbol{\Theta})$ which is defined as the summation of all differences between the two sides of the inequalities in Equation (3):

$$\begin{aligned} & S_{\text{Rank}}(D, D'|\boldsymbol{\Theta}) \\ &= \sum_{(a^*,q),(a,q)\in D} (F(v_r, v_q, v_{a^*}) - F(v_r, v_q, v_a)) \\ &+ \sum_{(a,q)\in D,(a_n,q)\in D'} (F(v_r, v_q, v_a) - F(v_r, v_q, v_{a_n}))\,, \end{aligned} \tag{3}$$

We only select (a, q) pairs from D and D' since (a, q) pairs provide sufficient coverage over all question instances for training in the CQA datasets.

Optimization

Cost Functions. We need to optimize the parameter $\boldsymbol{\Theta}$ that contains four parts: all embeddings of question raisers, all embeddings of question answerers, the parameters of the LSTM, and the parameters of the CNN-based scoring component. Since the optimal $\boldsymbol{\Theta}$ should maximize both Equation (1) and Equation (3), we alternatively maximize the two objective functions by gradient-based algorithms and back-propagation.

When NeRank converges, the embeddings and deep models in the optimum state have the following properties: (1) The entity embeddings contain proximity and expertise information to achieve personalized question routing; (2) The LSTM question encoder maps the content of new questions to a latent space where two additional aspects of information (expertise and proximity) are assessable in addition to text semantics. (3) The CNN recommender generates ranking scores using all three entity embeddings to measure the scores of answerers providing the "accepted answer".

Complexity. Suppose that single LSTM and CNN computations have T_1 and T_2 atomic operations respectively, a training batch has b instances, and the embedding dimension is d. The forward time complexity is $O(b(T1 + T2 + bd))$ per iteration.

Avoid Overfitting. We have the following mechanisms to prevent overfitting from happening on LSTM and CNN. (1) The LSTM is simplified to single-directional and single-layer to prevent over-parameterization; (2) Early stopping is utilized so the training terminates when the losses reach plateaux, which is shown in Figure 5; (3) The two objective terms are alternatively optimized towards different directions. They function as each other's regularizer that avoids overfitting.

Evaluation

In this section, we introduce the experiment settings, show experimental results, and demonstrate effectiveness and efficiency of NeRank.

Data and Experiment Settings

Two datasets of two real-world CQA websites with specific topics are employed to evaluate NeRank: Biologyand English. Each dataset[2] contains all questions raised before December, 2017 and all users' historical asking and answering records. The datasets differ in sizes (see Table 2) and are mutually exclusive in topics so that NeRank can be comprehensively tested. Other CQA datasets, such as Yahoo! Answers, are not selected for evaluation since they do not provide "accepted answers" that are needed to serve as the ground truth.

Table 2: Statistics of the datasets. r, q, and a represent question raiser, question, and question answerer.

Dataset	# of users	# of r	# of q	# of a
Biology	5,071	3,696	2,224	21,613
English	35,713	19,743	22,753	209,543

CQA networks are built from 90% of questions and the corresponding users to generate training walks. The rest 10% of questions and the corresponding raisers and answerers for testing. Users in the test set should have at least 5 asking or answering records to avoid cold starts. In each test query $\gamma = (r, q)$, we create a candidate answerer set of 20 answerers that includes all answerers of q in the dataset and

[2]Available at: https://archive.org/details/stackexchange

some other users randomly selected from the top 10% most responsive users. We choose the answerer with the highest predicted ranking score as the recommendation. The owners of the "accepted answers" are the ground truth.

The walks are generated from metapath "$AQRQA$" with the default length of 13 (three cycles) and the default node coverage of 20 (each node is covered at least 20 times in walk generation). The window size of Skip-gram model is set as 4; we use 3 negative samples per positive sample; and the dimension of learned embeddings is set as 256. We use a 64-channel CNN for ranking and the 300-dimensional *GoogleNews* pretrained word2vec model[3] to build the embedding matrix $\boldsymbol{x}$ for questions. NeRank is prototyped by Python 3.6.4 and PyTorch 0.4.0[4]. All experiments are conducted on a single 16GB-memory Tesla V100 GPU in an 512GB memory Nvidia DGX-1.

Results

This section reports the experimental results and analyses of the effectiveness and efficiency of NeRank. Note that three metrics, including Mean Reciprocal Rank (*MRR*), Hit at K (*Hit@K*), and Precision at 1 (*Prec@1*), are applied to evaluate the ranking performance.

Effectiveness of NeRank We compare NeRank with three baseline models shown as below.

- Score: A trivial method that recommends the answerer that has the largest number of "accepted answer".
- NMF: Non-negative Matrix Factorization (Gemulla et al. 2011) uses matrix decomposition to solve the ranking problem.
- L2R: SVM-based and RankingSVM-based learning to rank algorithms (Ji and Wang 2013) that extract features from user-question relations to predict the ranking.

The performances of NeRank and the baselines are shown in Table 3. NeRank significantly outperforms all baseline algorithms on both datasets in terms of all metrics. On the Biology dataset, NeRank achieves a Prec@1 of 0.387 and a Hit@K of 0.806, meaning that around 38.7% of the predictions are correct and the ground truth can be found in the top-5 ranked answerers in around 80.6% of the predictions. On the English dataset, NeRank achieves similar performances that the Prec@1 is 0.372 and Hit@k is 0.833. MRR in both dataset are around 0.56 indicating a huge improvement over the baselines in terms of overall ranking performance. All improvements of NeRank over the best baseline, NMF, are significant at 99% confidence in a *paired t-test*.

Some entries in Table 3 show that Biology has better results than English. Although, generally speaking, larger training sets may lead to better performance, the properties of the datasets may also play a role. In Biology, there exist a small group of proficient users with particular expertise. However, the English community has a larger proportion of skilled users since language is a common knowledge. Therefore, the performance may show small variance across datasets.

In summary, NeRank has a strong ability in discovering experts that provide the "acceped answer". The advantages of NeRank lie in the following facts: (1) NeRank considers question raiser information in addition to question content and question answerer, which can better conduct Question Routing. (2) NeRank utilizes deep neural network models that preserve the complex information of text semantic features and entity correlation features.

Table 3: The comparisons of MRR, Hit@K, and Prec@1 between NeRank and three baseline models.

Dataset	Biology			English		
Metric	MRR	Hit@K	Prec@1	MRR	Hit@K	Prec@1
Score	0.27	0.412	0.105	0.203	0.379	0.065
NMF	0.375	0.643	0.177	0.458	0.737	0.225
L2R	0.169	0.158	0.050	0.101	0.058	0.024
NeRank	0.563	0.806	0.387	0.567	0.833	0.372

Effectiveness of Metapath-based Embeddings We compare NeRank with two of its variants that employs, instead of metapath-based HIN embedding model, Deepwalk (Perozzi, Al-Rfou, and Skiena 2014) (denoted by "NeRank-DW") and LINE (Tang et al. 2015) (denoted by "NeRank-LINE") for embedding learning. Other configurations remain unchanged. We show that metapath benefits represention learning on HINs and improve the performance of NeRank.

Figure 4 shows the experimental results. We observe that NeRank achieves better results than NeRank-DW and NeRank-LINE on both datasets on all metrics. The reason is that Deepwalk and LINE are designed for homogeneous networks whereas CQA networks are heterogeneous that contain rich semantic information in diverse node and edge types. Metapath-based models take advantage of the semantic information and thus helps enhance the performance.

Although not particularly designed for HIN, Deepwalk and LINE are also capable of discovering the proximity relations between entities since both the metapath-based model and homogeneous network embedding models assume that connected entities have similarity. This accounts for the insignificance of the performance drop.

Effectiveness of Scoring Function We compare NeRank with another variant that replaces the CNN scoring function by $v_a \cdot \frac{v_r+v_q}{2}$, another combination of query `(question raiser, question content)`. The dot product of v_a and the numeric average of v_r and v_q is considered as the ranking score. Other settings are the same. We illustrate that our scoring function can effectively extract the latent expertise information and accurately generate ranking scores. The results are demonstrated in Figure 4 in which we denote the variant as "NeRank-AVG".

We observe that NeRank significantly outperforms NeRank-AVG by at least two folds. The performance difference is maximized in Prec@1 where NeRank-AVG can only correctly predict for 6.05% of the queries on Biology dataset and 0.04% on English dataset.

The huge performance gap indicates the strong ability of the CNN scoring function to capture the expertise informa-

[3]Available at: https://code.google.com/archive/p/word2vec/

[4]Available at: https://github.com/zyli93/NeRank

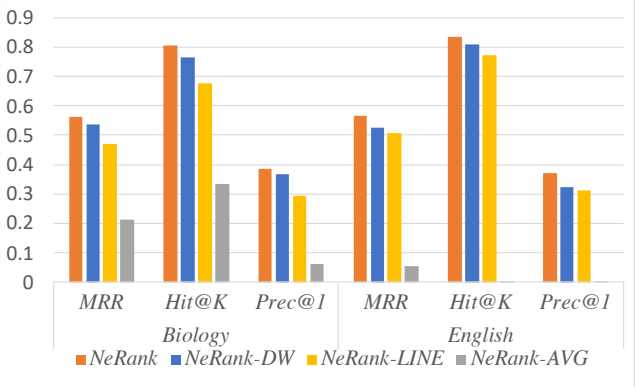
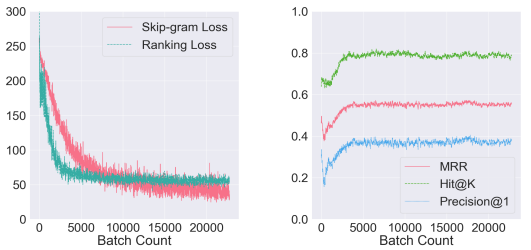
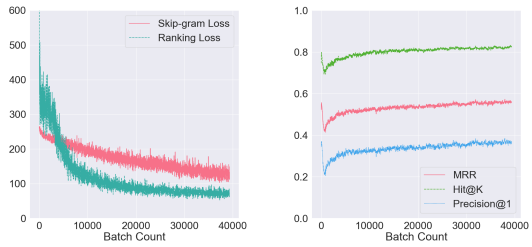


Figure 4: Performance comparison between NeRank and three variants.

tion from the correlations of entity representations and make accurate predictions.



(a) Losses on Biology (b) Metrics on Biology



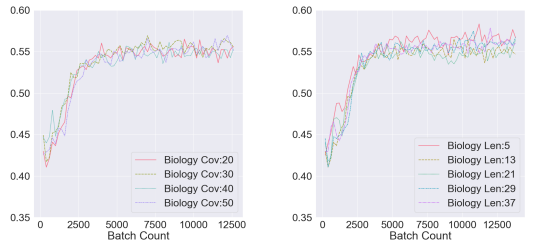
(c) Losses on English (d) Metrics on English

Figure 5: 5a and 5c show the decrease of two losses (negative objectives) with regard to iterations. 5b and 5d show the trends of three metrics with regard to iterations.

Convergence rate of NeRank We plot the trends of losses, i.e., negative objectives (Skip-gram objective in Equation (1) and Ranking objective in Equation (3)), and metrics as a function of the training iterations in Figure 5. These trends give us insight to the convergence rate of NeRank. These experiments are run in the default configurations.

It is observed that the NeRank converges at around 5,000 iterations (batch count) on the Biology dataset and at around 10,000 iterations on the English dataset. The convergences of the metrics happen before 5,000 iterations on Biology and before 10,000 iterations on English. Such converge rate is fast given the complex CNN and LSTM hybrid architecture of NeRank and a single GPU core, which demonstrates the model's learning efficiency and scalability. Convergences happen earlier on smaller dataset (Biology) and later on larger dataset (English). The reason is that, with the same batch size, a larger proportion of entities in smaller networks participates in training due to negative sampling.

Parameter Sensitivity We also evaluate the sensitivity of NeRank to node coverage and walk length. Node coverage refers to the number of times a certain node is covered by the training walks.We report the trends of MRR in Figure 6.



(a) MRR, Cov (Biology) (b) MRR, Len (Biology)

(c) MRR, Cov (English) (d) MRR, Len (English)

Figure 6: MRR trends over iterations on different node coverage (Cov) and walk length (Len).

It is observed that the curves almost coincide in the four subfigures, meaning that NeRank converges to very similar states at a similar speed although given different node coverages and walk lengths. Therefore, NeRank is robust to the changes of these hyperparameters.

Conclusion

In this paper, we propose NeRank, a framework for personalized question routing based on the question content and question raisers. NeRank learns representations of entities by heterogeneous network embedding and LSTM. Using the embeddings, the convolutional scoring model computes the ranking scores to predict the answerer that most probably contribute the "accepted answer". We test NeRank on two real-world CQA datasets. NeRank achieves a high performance and outperforms the state-of-the-art models.

Here we list the directions for future work (1) NeRank is unable to properly embed new onboarding users that have few asking or answering records. A possible solution is to learn mapping from his/her profile to an initial representation and optimize afterwards; (2) We may consider integrat-

ing historical answers that are also good information source for the answerer representation learning procedure.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. This works was partially supported by NIH U01HG008488, NIH R01GM115833, NSF III-1705169, NSF CAREER Award 1741634, Snapchat gift funds, and PPDai gift fund.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Chang, S., and Pal, A. 2013. Routing questions for collaborative answering in community question answering. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 494–501. ACM.

Chen, T., and Sun, Y. 2017. Task-guided and path-augmented heterogeneous network embedding for author identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 295–304. ACM.

Dong, Y.; Zhang, J.; Tang, J.; Chawla, N. V.; and Wang, B. 2015. Coupledlp: Link prediction in coupled networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 199–208. ACM.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 135–144. ACM.

Gemulla, R.; Nijkamp, E.; Haas, P. J.; and Sismanis, Y. 2011. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 69–77. ACM.

Gong, Y., and Zhang, Q. 2016. Hashtag recommendation using attention-based convolutional neural network. In *IJCAI*, 2782–2788.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864. ACM.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hearst, M. A.; Dumais, S. T.; Osuna, E.; Platt, J.; and Scholkopf, B. 1998. Support vector machines. *IEEE Intelligent Systems and their applications* 13(4):18–28.

Ji, Z., and Wang, B. 2013. Learning to rank for question routing in community question answering. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2363–2368. ACM.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Okura, S.; Tagami, Y.; Ono, S.; and Tajima, A. 2017. Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1933–1942. ACM.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.

Shi, C.; Hu, B.; Zhao, X.; and Yu, P. 2018. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Sun, Y., and Han, J. 2012. Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery* 3(2):1–159.

Sun, Y.; Han, J.; Yan, X.; Yu, P. S.; and Wu, T. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4(11):992–1003.

Sun, Y.; Norick, B.; Han, J.; Yan, X.; Yu, P. S.; and Yu, X. 2013. Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7(3):11.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, 1067–1077. International World Wide Web Conferences Steering Committee.

Zhao, Z.; Zhang, L.; He, X.; and Ng, W. 2015. Expert finding for question answering via graph regularized matrix completion. *IEEE Transactions on Knowledge and Data Engineering* 27(4):993–1004.

Zhao, Z.; Lu, H.; Zheng, V. W.; Cai, D.; He, X.; and Zhuang, Y. 2017. Community-based question answering via asymmetric multi-faceted ranking network learning. In *AAAI*, 3532–3539.

Zhou, C.; Sun, C.; Liu, Z.; and Lau, F. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Zhou, T. C.; Lyu, M. R.; and King, I. 2012. A classification-based approach to question routing in community question answering. In *Proceedings of the 21st International Conference on World Wide Web*, 783–790. ACM.