Expert2Vec: Experts Representation in Community Question Answering for Question Routing

Sara Mumtaz, Carlos Rodriguez, and Boualem Benatallah

School of Computer Science and Engineering, UNSW Sydney, NSW 2052, Australia
{s.mumtaz,carlos.rodriguez,boualem}@unsw.edu.au

Abstract. Communities of Question Answering (CQAs) are rapidly growing communities for exchanging information in the form of questions and answers. They rely on the contributions of users (i.e., members of the community) who have appropriate domain knowledge and can provide helpful answers. In order to deliver the most appropriate and valuable answers, identification of such users (experts) is critically important. However, a common problem faced in CQAs is that of poor expertise matching, i.e., routing of questions to inappropriate users. In this paper, we focus on Stack Overflow (a programming CQA) and address this problem by proposing an embedding based approach that integrates users' textual content obtained from the community (e.g., answers) and community feedback in a unified framework. Our embedding-based approach is used to find the best relevant users for a given question by computing the similarity between questions and our user expertise representation. Then, our framework exploits feedback from the community to rank the relevant users according to their expertise. We experimentally evaluate the performance of the proposed approach using Stack Overflow's dataset, compare it with state-of-the-art models and demonstrate that it can produce better results than the alternative models.

Keywords: Experts Finding · Question and Answering Communities· Embeddings Models· Stack Overflow· Expert Representation.

1 Introduction

Following the boom of web 2.0 technologies, there has been a dramatic increase in the number and variety of knowledge sharing communities. One such community is the Community Question Answering (CQA), which imparts knowledge in the form of questions and answers. Some of these communities are open to general discussions (e.g., Yahoo Answers[1]), whereas others (e.g., Stack Overflow (SO)[2]) are specialized communities focusing on technical knowledge (SO, in particular, focuses on programming knowledge). The goal of these communities is to provide accurate and timely answers to questions posted by users of the platform

[1] https://answers.yahoo.com/
[2] https://stackoverflow.com/

The final authenticated version is available online at
https://doi.org/10.1007/978-3-030-21290-2_14

supporting the communities. Achieving this goal entails the recommendation of the question to the right user (answerer) for which knowledge and expertise in the question's topic is essential. The process that identifies such user is called Expert Identification [22] and involves finding a skillful and knowledgeable user that can answer a given question. Unlike general CQAs, specialized communities not only require more technical skills but also demand various levels of expertise on specific topics in order to provide answers with long lasting relevance for the community [1].

Despite the success, CQAs face a number of problems and challenges that are still far from being solved. Among them, poor expertise matching [21] is one of the most significant problems. Poor expertise matching implies that a new question may not be matched with the appropriate user who possesses the expertise to correctly answer the question. This may result in answers either being given incorrectly, scammed or not given at all [19]. To address this problem, a wide range of models have been developed, among them the well known probabilistic language models [3], which estimate users expertise from their relevant documents based on co-occurrences. Here, the answers provided by users are considered to be the reflection of their expertise [16]. Although, these approaches have achieved reasonable performance, they are prone to term mismatch problems [11] between search queries and documents (or questions and experts, in the context of CQAs). Topic models, such as Latent Dirichlet Allocation (LDA) and its variants [4], have been introduced thereafter to overcome this problem by capturing the latent semantic relationship between users and their queries (see, e.g., [21]). These approaches, however, do not perform well when dealing with short-text documents (which is typically the case for posts in CQAs) where only limited word co-occurrence instances can be found [20].

In order to address these limitations, in this paper, we focus on the specialized community of SO and propose a framework for better representation of user expertise that leverages on word embedding techniques [17]. This framework not only makes use of textual content as evidences of user expertise within the community, but it also leverages the community feedback and interactions of users with one another to better match user expertise for routing questions. In a nutshell, the contributions of this paper are as follows:

- We propose an effective framework for representation of user expertise via word embedding techniques applied on the content of their contributions. We furthermore leverage on community feedback in order to rank users based on their contributions and performance within the community.
- We build our own word embedding model from SO's corpus to better capture domain-specific semantics within this community.
- We evaluate the performance of our proposed solution through experiments on SO's dataset and by comparing it with state-of-the-art models.

The rest of the paper proceeds as follows. The problem statement and the characteristics of SO platform is introduced in Section 2. Section 3 explains the framework for our proposed approach. The experiments and evaluations are presented in Section 4. Related work is given in Section 5, and, finally, Section 6 concludes the paper with ideas for future work.

2 Preliminaries

This section formally presents the problem statement and provides background information to better understand the different Q&A elements and mechanisms in SO's platform.

2.1 Problem Statement

Given a question Q, our goal is to find a set of experts $\mathcal{U}$ that best match to the required skills and domain knowledge to answer Q.

Definition 1. *Experts in CQA [16]. Given a new question Q on topic X, a user can be considered as an expert to answer Q if he/she has already answered a set of similar questions $Y = \{y_1, y_2, ..., y_z\}$ in the past. In addition, his/her expertise can also be inferred from his/her behavior and interactions within the community.*

More specifically, if we are given a new question Q and n number of users $U = \{u_1, u_2, ..., u_n\}$ having a set of m answer posts $P = \{p_1, p_2, ..., p_m\}$, our goal is to find a set of k users $\mathcal{U} = \{u_1, u_2, ..., u_k\}$ capable of answering Q. To be aligned with the definition of experts in CQA, we consider the users' answers as an indicator of their expertise on the question's topic and estimate their performance through community feedback.

As we will explain in the following section, our work leverages on users' textual content (i.e., answers) to match questions and experts. Therefore, a key *challenge* we face is the representation of user expertise for matching the newly posted question and users (experts) using a content-based method [16]. Given that SO is a focused programming community, unlike general CQAs (such as Yahoo Answers), it requires sufficient technical expertise for a user to answer a question. Keeping this in view, the second challenge entails quantifying the level of expertise of users.

2.2 Stack Overflow

SO is an interactive knowledge sharing platform in the area of computer programming and software development. It managed to get the attention of a large audience with nearly 8 million users and 41 million posts (25 million answers and 16 million questions)[3]. Users in SO interact by asking and answering questions as well as editing and commenting on posts in order to curate content. Similarly, they acknowledge the most helpful answers by up/down voting and expressing their opinion on the quality of posts. SO also incorporates a reputation system that reflects the users' activities and their engagement toward contributions that encourage users' participation.

As an example, Figure 1 illustrates the different elements of a Q&A post in SO. The asker posts a question that consists of three parts: (i) title, (ii) body and (iii) tags. The title of a question is a short description of the essence of the question, whereas the body explains the question in more details. A tag (which can be considered a metadata) provides a glimpse into the topic of the question.

[3] https://insights.stackoverflow.com/survey/2018/

Fig. 1. An example of Stack Overflow Q&A post.

The example question in Figure 1 has two tags, *JavaScript* and *URL*, which indicates that the question is related to JavaScript and URLs. Users provide different answers according to their knowledge and expertise (20 answers for our example question). Moreover, the asker can accept one answer among all answers, which is typically referred to as the best answer. Furthermore, other users can also up/down vote questions and answers. Such feedback mechanisms are built upon the important concepts of trust and reputation [1], which allow SO to associate reputation scores and badges to users[4].

3 Representing Users Expertise and Recommending Users to Questions

This section presents our proposal for representing user expertise and recommending users to questions. Given the nature of the contributions in SO (which is made mainly of textual content), the key intuition behind our proposal is that we can leverage on such contributions to derive a suitable representation for user expertise. As we will discuss next, we leverage on state-of-the-art techniques to embed such expertise into a vector space that can be later used to match users with questions. The framework for our proposed solution is outlined in Figure 2 and consists of two main phases:(i) *user expertise representation*, and (ii) *user recommendation*. We discuss the details of each of these phases next.

3.1 User Expertise Representation

A key requirement for a good matching of user expertise and a new question consists in having a good representation of the expertise of users participating in a CQA. In this phase, users' answers and their expertise are represented by using embedding models [14], which form the basis for capturing the semantic relationship between a new question and users expertise. Word embeddings stem

[4] https://stackoverflow.com/help/whats-reputation

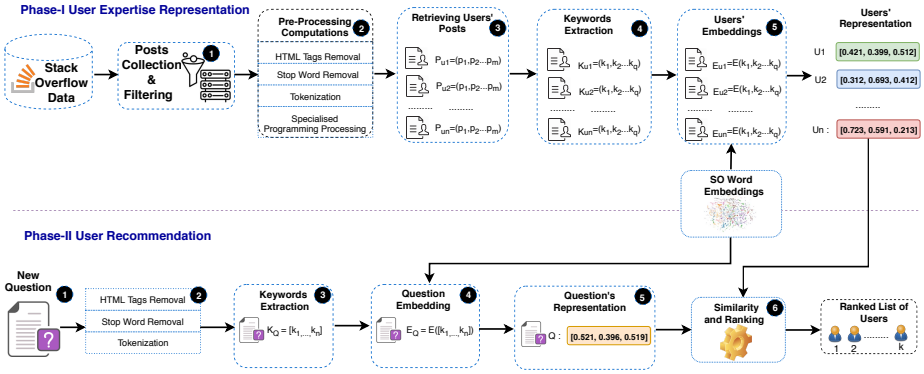

Fig. 2. Overall framework for our proposed solution.

from the distributional hypothesis [17], which states that words occurring in similar contexts tend to have similar meanings. This is a powerful property that have been leveraged in the literature to develop robust semantic similarity techniques in various domains [8]. In the following, we describe step-by-step how our framework (see Figure 2) leverages on this property for user expertise representation in CQAs.

Posts Collection and Filtering. The first step in our framework involves the collection of Q&A posts from SO. The collected posts are filtered based on two requirements: (i) already resolved posts (i.e., question with answers), and (ii) having accepted (best) answer marked by the question poster. The purpose of this early filtering is to filter out low quality posts and reduce noise in the dataset.

Pre-Processing Computations. Pre-processing is an essential step that transforms raw text into a format suitable for further processing. In Step 2 of our framework, we apply standard Natural Language Processing (NLP) techniques[5], which include HTML tags removal, tokenization, stop words/frequent words removal and lemmatization. Apart from standard text processing, we also take special consideration to retain the symbols that are meaningful in the context of programming such as & and $. For example, the symbols associated with C++ and C# are key to differentiate the two programming languages.

Retrieving Users' Posts. In our approach, our assumption is that users posting answers to questions can be considered knowledgeable on the question's topic. We therefore represent users' expertise in CQAs through their answers. In order to do so, after pre-processing, Step 3 retrieves posts from each user (see, e.g., $P = \{p_1, p_2, ..., p_m\}$ in Step 3 of Figure 2). Here, the selection of users/answers are also based on satisfying specific criteria, i.e., (i) users with at least one best (accepted) answer, and (ii) answers with score (upvotes - downvotes) of 5 or

[5] https://www.nltk.org/

more (users with high expertise get high score/votes for their answers [7]). This step filters out users with low quality posts.

Keywords Extraction. Step 4 involves taking answer posts from users and decomposing them into a set of keywords $\{k_1, k_2, ..., k_q\}$ with the help of a part-of-speech tagger[6]. It is important to note that, unlike other CQAs, users' answers from programming CQAs usually contain massive code snippets. Filtering out such codes snippets during the pre-processing stage results in a very sparse set of keywords for each post. In order to deal with this problem, we combine the keywords from a post with tags (associated to the post) and their description for better representation (tags in SO come with their own description). This enriched set of keywords is then used to represent users in an embedding space as explained next.

User Embeddings. Following keywords extraction, Step 5 in our framework consists in representing users via these keywords with proper semantic embeddings/vector representation, which will be used in the next phase to capture the semantic similarity between a question being asked and users' expertise. The vector representation can be computed in various ways. One popular technique is the vector space model [2], which considers users' answers as a bag of words and uses occurrences count of words as features. A typical problem with this model is that it classifies semantically similar words such as *car* and *automobile* differently [26]. To avoid this problem, we employ embedding-based models that utilise the (contextual) co-occurrences of words to represent them in a lower dimensional vector space and capture their contextual and semantic relationship.

More specifically, to compute the vector representation $\overrightarrow{U}$ for each user expertise, we first learn a vector representation of words from SO's corpus using Word2Vec Skip-gram model [17]. The Skip-gram model uses a neural network architecture to learn distributed word embeddings. This enables the prediction of the surroundings of words in associated context(s) from a corpus. We trained this model on SO's full dataset (more details is given in experimental section) to obtain domain-specific word embeddings v_w for each word w in SO's corpora. Using this word embedding model, denoted by $\mathcal{WE}$, we then generate each user expertise embedding $\overrightarrow{U}$. To do so, we make use of *vectors averaging* and *weighted vectors averaging* as discussed next.

Vectors Averaging (VecAvg). A simple yet powerful way for text representation through embeddings consists in aggregating its individual word embeddings through sum or averaging [8]. In this work, using $\mathcal{WE}$, we compute the average of the word vectors of each keyword $k_i \in K_{ui}$ in the user's answer post to obtain the semantic representation of users' expertise. That is, for each user u_j, we compute:

$$\overrightarrow{P_m} = \frac{1}{q} \sum_{i=1}^{q} e(k_i) \tag{1}$$

where $\overrightarrow{P_m}$ is the vector representation of user u_j's answer post, q is the number of keywords obtained from the post and $e(k_i)$ is a function that obtains the

[6] https://nlp.stanford.edu/software/tagger.shtml

embedding of keyword $k_i \in K_{uj}$ from $\mathcal{WE}$ (notice that, for simplicity, the right hand side of Eq. (1) is represented as function $E(k_1, k_2, .., k_q)$ in Step 5 of Figure 2). Similarly, the vector representation $\overrightarrow{U_j}$ for users can be obtained by averaging the vectors of all their answer posts.

Weighted Vector Averaging. The vector averaging method above considers all keyword in an answer post as equally important, which leads to vector representations of users' expertise that ignore the role that certain keywords play within an answer. To address this limitation, we use the weighted vector averaging method. Here, weights are assigned to each keyword to reflect their importance in an answer post. In this work, these weights are determined by the IDF (Inverse Document Frequency) weighting scheme [6]. The IDF is computed as the inverse of the number of user posts that contain a specific keyword, i.e.,

$$IDF_k = log\frac{m}{PC_k} \tag{2}$$

where IDF_k is the IDF of keyword k, m is the total number of answer posts of a user u_j and PC_k represents the number of answer posts containing keyword k. Thus, the weighed vector averaging not only leverages on the distributional semantics of word embedding techniques but also takes into consideration the information about the distribution of keywords appearing in different user posts. The actual computation of a users' expertise representation ($\overrightarrow{U}$) using this method can be obtained by weighting the embedding of each keyword k in Eq. (1) by the corresponding IDF_k obtained through Eq. (2).

3.2 User Recommendation

This phase of our framework aims to route a newly posted question to users that are expert on the topic of the question. We achieve this in two steps: (i) *similarity computation*, and (ii) *user ranking via statistical features.*

Upon receiving a new question Q, phase-II performs similar pre-processing and keyword extraction steps as given in phase-I. Then, it leverages $\mathcal{WE}$ to generate a vector $\overrightarrow{Q}$ for the given question using the vector averaging method, which is used for similarity computations as detailed next.

Similarity Computations. After learning the vector representation of users' expertise $\overrightarrow{U}$ and question $\overrightarrow{Q}$, we compute the similarity score $sim(Q, U)$ via cosine similarity [26], which is defined as follows:

$$sim(Q, U) = \frac{\overrightarrow{Q}.\overrightarrow{U}}{|\overrightarrow{Q}||\overrightarrow{U}|} \tag{3}$$

The closer the value of the cosine similarity to 1, the greater the relevancy of the new question with the users' expertise. This similarity computation is done for each user u_j, which results in a ranked list of users U_s, such that $U_s \subset U$ and:

$$U_s = sim(Q, U) \tag{4}$$

Here, the rank in U_s is determined by the results of the cosine similarity. Next, we show how we employ statistical features to further improve this ranking.

Users Ranking via Statistical Features. In addition to domain knowledge, the expertise of users can also be inferred from their activities and behaviour in CQAs. These activities comprise different signals based on user performance, reputation and impact in the community. Table 1 presents a list of statistical features from SO that can be used as expertise indicators. We consider all these signals and features in our work as given in [12]. The key intuition behind this choice is that expertise can be defined collectively by a set of indicators on the platform that capture the characteristics of users indicating their expertise on a given topic. We elaborate more on these features next.

Activity features. These are features related to users' activities on the platform. Prior research [10] has shown that this set of features is useful in finding experts. For example, the best answers measure is an indication of a user being an expert in that domain area. Another example includes the z-score $z(q,a) = \frac{a-q}{\sqrt{a-q}}$ measure introduced in [23], where q and a are the number of users questions and answers, respectively. This score combines users' asking and replying patterns with the intuition that asking a lot of questions often indicates the lack of expertise on some topics and vice versa.

Community features. This set of features shows how much the community trusts an answerer. Feedback from the community indicates the quality of users' contributions [7]. Intuitively, users with high reputation get more votes for their posts as compared to others. Similarly, badges are given to users to reward them for their contributions once they reach a certain threshold.

Time-Aware features. We also include this feature to observe user activities over a period of time. One sample feature in this category includes the days between a user a joining the community and his/her n-th best answer.

Table 1. Set of features comprising different signals from platform.

Features Set	Features
Activity(A)	Number of questions, number of answers and number of accepted answers, z-score computation
Community(C)	Number of badges/awards, number of page views, number of favourites, total reputation and average answers score
Time-Aware(T)	Time interval (days between joining and n-th best answers) and average answers time

We leverage on the statistical features introduced previously and apply different linear combinations of these features. More formally, the aim is to rank a user with respect to his/her level of expertise and knowledge through his/her performance in the community. To do so, an expertise score $exp(Q, U_s)$ of each user in U_s for a question Q is computed as follows:

$$exp(Q, U_s) = \alpha(A) + \beta(C) + \gamma(T) \tag{5}$$

where α, β and γ are weights that range from 0 to 1. These weights distribute the importance of activity, community and time-aware features in determining the expertise of users.

We briefly describe the usage of these features in our work next. The level of user expertise is derived from Eq. (5) by aggregating the features given in Table 1. A is an aggregation of user *activity features*. More concretely, $A(Q, U_s) = x + y$

where x computes the acceptance ratio of a users' answers, i.e., $x = \frac{B(A_Q)}{A_Q}$. Here, $B(A_Q)$ represents the number of a user's *best* answers and A_Q is the total number of his/her answers. Similarly, y quantifies the z score ($\frac{a-q}{\sqrt{a-q}}$) introduced previously. Intuitively, a high z score indicates a high knowledge of the user of the topic related to Q. Likewise, *community features* represent the influence of a user in the community and it is expressed as $C(Q, U_s) = p + q + r + s$. Where p represents the number of badges a user received, q combines the number of favourites with number of page views of a user's posts, r computes user reputation relative to the overall reputation in the community, and s is the average answer score of a user. The *time-aware feature* (T) is another factor that indicates user expertise over a period of time, with the intuition that expert users take less time in answering questions. Here, $T(U_s) = t_{ba} + t_{avg}$, where t_{ba} is the number of days between user joining the community and the n-th best answers, and t_{avg} is the average answer time of a user. The less time the user takes in answering questions, the greater the expertise of the user.

Recommendation Score. The last step in our framework is to recommend an appropriate set of users to answer a question Q. Hence, a recommendation score is calculated based on the user relevance and expertise to Q. More precisely, we integrate the Eq. (4) and (5) to get the final ranking of users as:

$$R(Q, U_s) = \psi(U_s) + (1 - \psi)exp(Q, U_s) \tag{6}$$

The first term in Eq. (6) deals with the contribution of a user relevance to a question Q whereas the second term is the contribution of user expertise in relation to Q. Here, ψ is a weight in the range $[0, 1]$ showing the importance of the terms in Eq. (6) in recommending expert users for answering Q. In the end, this step returns $top-k$ expert users in decreasing order of their recommendation score, i.e., $\mathcal{U} = \{u_1, u_2, ..., u_k\}$. Here, the user with the highest score (i.e., best ranked) is the best user to answer the given question.

4 Experiments

This section evaluates the performance of our proposed approach through a set of experiments with the aim of answering the following research questions:

***RQ1*:** How effective is our proposed approach for experts representation via embedding models in question-to-expert matching?

***RQ2*:** What is the impact of the statistical features employed in our approach in improving question-to-expert matching?

4.1 Experimental Setup

We run our experiments using real data from SO. In the following, we provide the details of our experiments, which includes data preparation, parameters setting along with models and metrics used for comparison purposes.

Data Preparation. We first start with the details regarding data preparation. We describe next the dataset, ground truth and test data used for experimentation purposes.

Dataset. The subset of data is obtained from SO's dataset[7] covering the period from August 2008 to August 2018, which consists of posts tagged by Java and

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

its co-occurred tags (*Java Script, HTML, etc.*). Java tag has the highest ratio of Q&A posts in our dataset, and thus covers a large amount of data with 3,345,038 posts. As stated earlier in Section 3, the selection of users and posts are based on certain requirements: Firstly, we prune questions with no answers, which results in 2,904,208 posts. Secondly, we extract only the stabilised posts. Stabilised posts refer to resolved questions which have an accepted (best) answer. Similarly, users are also selected based on two criteria: (i) the user has at least one best answer, and (ii) his/her reputation score in the community must be more than the average reputation in the community. Furthermore, only answer posts with a score of 5 or more are selected. In the end, our dataset consists of 778,418 posts and 54,851 users.
Ground Truth. As for the ground truth for our approach, we make use of heuristics instead of manual labeling, as the latter does not scale well with the sheer amount of data. Following the experimental settings given in [10] for ground truth, we also define users expertise based on their accepted (best answers). A user is considered as an expert if: (i) he/she has at least ten or more accepted (best) answers, and (ii) his/her answers' acceptance ratio as well as their reputation score must be higher than the average user ratio on the platform. Thus, following these heuristics, the ground truth is made of users that have shown expertise on the topics corresponding to the answered questions.
Test Data. We prepare two different test datasets to carry out our experiments. The first one consists in a collection of Java unanswered questions (so the training and test data do not overlap), while the second one comprises randomly collected questions related to Java with actual best answerers.

Parameters Setting. To obtain domain specific embeddings, we trained our own embedding model instead of making use of already available pre-trained model over general dataset such as Wikipedia[8]. For instance, the domain specific word "cmd" (typically, short for "command") might not be available in the general trained model. Similarly, a word such as "Python" has different contextual meaning in software engineering vs. other domains. The parameters required to train this model are set as follows: The vector dimension to 200 and a sliding window for word co-occurrences to 5. In addition, a minimum word count set to 20 means that a word must occur at least 20 times in the training set. This setting of parameters gave best performance in term of matching accuracy.

Environment and Implementation. This work is implemented in Python 3 and experiments are conducted on a Linux machine with Intel Xeon 2.80GHz CPU and 16GB memory.

Performance Comparison. In order to demonstrate the effectiveness and matching accuracy of our approach, we compare it with the following models.
Language Model (LM) [3]. It is a classical language model for expert finding, which considers documents associated with users as their expertise. We use Jelinek Mercer (JM) smoothing with parameter $\lambda = 0.25$ for experiment purposes.
Latent Dirichlet Allocation (LDA) [4]. LDA is an unsupervised generative probability model that represents documents into a latent semantic space and then extracts topics. We use LDA to extract the topic vectors of users' answers and then compute the cosine similarity to match with the given question. The number of topics is set to 100 for this purpose.

[8] https://github.com/3Top/word2vec-api

Latent Semantic Indexing (LSI) [9]. LSI is essentially a bag of words and is used to identify the similarity between queries and user expertise.
Document Representation (Doc2vec) [14]. Doc2vec is an extension of word2vec [17]. It is based on the distributed bag of words representations to extend the learning from word to word sequences. In this paper, we use the distributed bag of words idea to learn vectors directly by predicting all words in the answers. To be aligned with word level representation, we use the same hypermeters as of word2vec model with a sample size of $1e5$.
Word Movers Distance (WMD) [13]. WMD is the minimal cumulative distance between words of two documents. It retrieves word vectors from pre-trained word embedding model and considers each text document as a normalised bag of words. The weight of each word is considered as the number of times the word occurs in the document. The high occurrences show the importance of each word in that document. This approach does not require any hypermeters.

Evaluation Metrics. We evaluate the performance of our proposed solution based on one of the most used information retrieval measures for expert finding [3] and question routing in CQA [24], i.e., $P@N$(precision at N). It measures the percentage of relevant users/experts found at the top N retrieved results. More specifically, it is the ratio of the number of relevant users $@N$ to the total number of users until N. For example, if our system returns 10 users for a given Q, where the relevant users are ranked at 1,5,6 and 9. Then P@5 is $\frac{2}{5}$ in this case. We also use Mean Averaged Precision (MAP), which is the average precision of all precisions for all test questions. Besides this, a user can also be ranked via the metric Success@N ($S@N$) introduced in [15] for finding the best answerers. If a system returns the actual best answerer for a given question, then prediction is successful. Otherwise, it is not. For a test question, if the best answerer is among the top N experts returned by the system, then the value of the $S@N$ is the reciprocal rank (RR), i.e., $1/N$. From here, we can also calculate the Mean Averaged Reciprocal (MRR) across multiple questions as follows:

$$MRR = \frac{1}{|M|}\sum_{i=1}^{|M|}\frac{1}{Rank_i} \tag{7}$$

where $|M|$ is the number of questions and $Rank_i$ is the position of the best answerer in the ranked list returned by the system for the i-th question.

4.2 Results and Discussions

RQ1: Effectiveness of Embedding Models. The first set of experiments investigates how well users' expertise matched a given question using embedding models. The performance of different models in terms of precision$@N$ and MAP is summarised and reported in Table 2. It is interesting to note that the simple word level representation, i.e., *VecAvg*, achieved better performance over others in terms of recommending relevant experts. LSI and LDA have also achieved reasonable performance. Despite this, the embedding models outperformed both of them. Overall, results suggest that simple word level representation works better in representing user expertise and routing questions.

Table 2. Performance comparison of different methods.

Method	P@1	P@5	P@10	MAP
LM	0.36	0.34	0.36	0.43
LSI	0.40	0.36	0.30	0.53
LDA	0.50	0.36	0.34	0.45
Doc2Vec	0.60	0.41	0.38	0.56
WMD	0.50	0.44	0.35	0.56
VecAvg	0.60	0.48	0.42	0.62
Weighted VecAvg	0.60	0.45	0.44	0.63

RQ2: Effect of Statistical Features. The second research question explores the importance of statistical features in ranking a user. For this purpose, we evaluate the performance of our approach through MRR for best answerers prediction and list the results in Table 3. The $MRR(R)$ measure in Table 3 ranks users based on the relevance between their answers and questions (see Eq. (4)). Whereas $MRR(S)$ make use of statistical features and combines the user relevance with expertise to rank them accordingly (Eq. (6)). The larger the value for these metrics, the better the performance of the corresponding technique. It can be observed from the results that there is a small improvement in the ranking using the expertise indicators. These results show that the usage of expertise indicators improves user ranking, and thus, the prediction of best answerers for a given question.

Table 3. Best answerer prediction.

Method	MRR(R)	MRR(S)
LM	0.041	0.045
WMD	0.154	0.155
VecAvg	0.161	0.163

Discussions. The results reported in Table 2 and 3 show that expertise representation via embeddings indeed produce better results w.r.t. the more traditional techniques of LM, LSI, LDA and WMD. When considering P@1, the embedding models outperform their counterparts by approximately 20% (WMD) to 67% (LM). When comparing performances w.r.t. the usage (or not) of statistical features, we can see modest improvements across the representative models in Table 3, ranging from 1.2% (VecAvg) to 9.8% (LM). Overall, these results are promising and encouraging, and opens up new opportunities for further research into this direction.

The experiment reported in this work comes with its own *limitations*. Firstly, we used only a subset of SO's dataset (focused only on Java-related posts), which limits our ability to generalize our conclusions to the whole platform (and for all topics in SO). Secondly, given the lack of a labeled dataset for expertise in SO, we relied on heuristics to determine the expertise of users. While this is a best-effort attempt to cope with such issue, the heuristics we used have proven useful also in previous work [10]. Finally, we trained our word embedding

model (with Word2Vec) only on SO's dataset. More robust embeddings could be obtained by considering additional corpus from other sources such as Github[9] and programming-related wikis and forums.

5 Related Work

The concept of expert finding has been well-studied in information retrieval and knowledge management [22]. More recently, an increased attention has been given to the analysis of CQAs. As a result, many algorithms and models have been employed that can be categorised into three types: Graph based, content based and deep learning based approaches.

Graph based approaches. These approaches are based on graph analysis. They are built upon underlying interactions among users in the community [18] to understand their activities. An early work in this direction was introduced [23] for the online help seek community to find people who can provide helpful answers. For this purpose, they applied network algorithms such as PageRank [6] to estimate relative expertise of users from their interactions. Similarly, [5] proposed a model called InDegree that considers the number of best answers provided by users as an expertise indicator. The higher the degree of a user, the more likely he/she is an expert. These approaches, however, are based only on user interactions in the community rather than their textual contents as an evidence of their expertise. Thus, graph based approaches are helpful in finding the influential users in the community but not the topical expertise.

Content based approaches. Content based approaches usually consider expert finding as an Information Retrieval problem. For example, seminal works like [3] on probabilistic language modeling technique consider documents associated with users as the reflection of their expertise. In the context of CQAs, questions being asked can be considered as queries, while answers provided by users are considered the reflection of their expertise [16]. Here, the need for an exact overlap between query and document terms leads to a vocabulary gap problem. Topic modeling [4] based approaches, such as [21], address this problem by incorporating semantic similarity to discover the latent topics in contents of question and identifies the interests of users. Compared to language models, topic modeling achieves better performance in terms of retrieving expert users. Despite the improved performance shown by content based approaches, it is usually difficult to decide which features are to be used for a specific dataset. Moreover, these methods require more efforts and time due to the excessive dependence on handcrafted features. Therefore, more recently, increased attention has been given to deep learning techniques.

Deep Learning based approaches. Deep learning approaches have been widely used primarily for their effectiveness and promising results with little human involvement in feature selection and engineering. Work in the context of expert finding was introduced by [25], which leverages the textual contents and social interactions among users for better similarity matching. Their method utilises DeepWalk for social relations and Recurrent Neural Networks (RNNs) for modelling users relative quality rank to questions. Similar to this work, [24] proposed a framework, which models questions, answers and users as a heterogeneous network and trains a deep neural network on random walks to enhance

[9] https://github.com

matching. A very recent work [26] makes use of the Convolutional Neural Network (CNN) for expert identification in CQAs. Their approach combines question and user representations for finding questions that are similar to a given question. The user representations in their framework is build up using DeepWalk and word embeddings are used to represent questions. [19] proposes an approach that not only addresses the problem of matching degree between users' expertise with the query but also the likelihood of an expert to answer the query after inviting him/her to do so.

There is also a large body of work addressing the same problem in other professional communities such as LinkedIn[10]. For example, [12] focuses on determining the topical expertise based on the skills dataset available within the company. Similarly, expert finding is also studied in bibliographical information networks [26] to mitigate the vocabulary gap problem by introducing locally trained embedding with network ranking algorithm. Unlike the approaches above, in our work, we combine content- and embedding-based techniques along with statistical features stemming from both user activities and feedback provided by the community (crowd).

6 Conclusion

In this work, we apply the state-of-the-art embedding models for representation of users' expertise to address the problem of poor expertise matching in CQAs. The key of our approach is the use of user content as well as community feedback in formulating their expertise. Firstly, we introduce the embedding models that map the newly posted question and users' contents into low dimensional semantic space and then relevance between them is computed via cosine similarity. Secondly, we take the advantage of the community feedback from the platform and incorporate it as an expertise indicator that helps rank the relevant users according to their level of expertise. We performed extensive experiments on Stack Overflow's dataset and compared the results with previous models. The experimental results show that our proposed framework outperforms the previous models and baseline approaches in terms of routing questions to experts.
In future work, we would like to extend the proposed approach for expert representation to heterogeneous data sources combining, e.g., Stack Overflow, Github and other programming communities.

Acknowledgement. The work of the second and third authors is supported by Data to Decisions CRC (D2D-CRC).

References

1. Anderson, A., Huttenlocher, D., Kleinberg, J., Leskovec, J.: Discovering value from community activity on focused question answering sites: A case study of stack overflow. In: Proceedings of KDD '12. pp. 850–858 (2012)
2. Baeza-Yates, R., Ribeiro, B.: Modern information retrieval (2011)
3. Balog, K., Bogers, T., Azzopardi, L., de Rijke, M., van den Bosch, A.: Broad expertise retrieval in sparse data environments. In: Proceedings of SIGIR '07. pp. 551–558 (2007)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research **3**(Jan), 993–1022 (2003)

[10] https://www.linkedin.com

5. Bouguessa, M., Dumoulin, B., Wang, S.: Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In: Proceedings of SIGKDD'08. pp. 866–874 (2008)
6. Christopher, D.M., Prabhakar, R., Hinrich, S.: Introduction to information retrieval. An Introduction To Information Retrieval **151**(177), 5 (2008)
7. Dalip, D.H., Gonçalves, M.A., Cristo, M., Calado, P.: Exploiting user feedback to learn to rank answers in q&a forums: a case study with stack overflow. In: Proceedings of SIGIR'13. pp. 543–552 (2013)
8. De Boom, C., Van Canneyt, S., Demeester, T., Dhoedt, B.: Representation learning for very short texts using weighted word embedding aggregation. Pattern Recognition Letters **80**, 150–156 (2016)
9. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American society for information science **41**(6), 391–407 (1990)
10. van Dijk, D., Tsagkias, M., de Rijke, M.: Early detection of topical expertise in community question answering. In: Proceedings of SIGIR'15. pp. 995–998 (2015)
11. Gui, H., Zhu, Q., Liu, L., Zhang, A., Han, J.: Expert finding in heterogeneous bibliographic networks with locally-trained embeddings. CoRR **abs/1803.03370** (2018)
12. Ha-Thuc, V., Venkataraman, G., Rodriguez, M., Sinha, S., Sundaram, S., Guo, L.: Personalized expertise search at linkedin. In: Big Data (Big Data), 2015 IEEE International Conference on. pp. 1238–1247. IEEE (2015)
13. Kusner, M., Sun, Y., Kolkin, N., Weinberger, K.: From word embeddings to document distances. In: ICML'15. pp. 957–966 (2015)
14. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: ICML'14. pp. 1188–1196 (2014)
15. Liu, M., Liu, Y., Yang, Q.: Predicting best answerers for new questions in community question answering. In: International Conference on Web-Age Information Management. pp. 127–138 (2010)
16. Liu, X., Croft, W.B., Koll, M.: Finding experts in community-based question-answering services. In: CIKM'05. pp. 315–316 (2005)
17. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
18. Mumtaz, S., Wang, X.: Identifying top-k influential nodes in networks. In: CIKM'17. pp. 2219–2222 (2017)
19. Qian, Y., Tang, J., Wu, K.: Weakly learning to match experts in online community. In: IJCAI'18. pp. 3841–3847 (2018)
20. Quan, X., Kit, C., Ge, Y., Pan, S.J.: Short and sparse text topic modeling via self-aggregation. In: IJCAI'15 (2015)
21. Yang, L., Qiu, M., Gottipati, S., Zhu, F., Jiang, J., Sun, H., Chen, Z.: Cqarank: Jointly model topics and expertise in community question answering. In: Proceedings of CIKM'13. pp. 99–108 (2013)
22. Yimam-Seid, D., Kobsa, A.: Expert-finding systems for organizations: Problem and domain analysis and the demoir approach. Journal of Organizational Computing and Electronic Commerce **13**(1), 1–24 (2003)
23. Zhang, J., Ackerman, M.S., Adamic, L.: Expertise networks in online communities: Structure and algorithms. In: WWW'07. pp. 221–230 (2007)
24. Zhao, Z., Lu, H., Zheng, V.W., Cai, D., He, X., Zhuang, Y.: Community-based question answering via asymmetric multi-faceted ranking network learning. In: AAAI'17. pp. 3532–3539 (2017)
25. Zhao, Z., Yang, Q., Cai, D., He, X., Zhuang, Y.: Expert finding for community-based question answering via ranking metric network learning. In: IJCAI'16. pp. 3000–3006 (2016)
26. Zheng, C., Zhai, S., Zhang, Z.: A deep learning approach for expert identification in question answering communities. CoRR **abs/1711.05350** (2017)