Question Routing to User Communities

Aditya Pal, Fei Wang, Michelle X. Zhou, Jeffrey Nichols, Barton A. Smith
IBM Research Almaden, USA
{apal, wangfe, mzhou, jwnichols, barton.smith}@us.ibm.com

ABSTRACT

An online community consists of a group of users who share a common interest, background, or experience and their collective goal is to contribute towards the welfare of the community members. Question answering is an important feature that enables community members to exchange knowledge within the community boundary. The overwhelming number of communities necessitates the need for a good question routing strategy so that new questions gets routed to the appropriately focused community and thus get resolved. In this paper, we consider the novel problem of routing questions to the right community and propose a framework to select the right set of communities for a question. We begin by using several prior proposed features for users and add some additional features, namely language attributes and inclination to respond, for community modeling. Then we introduce two k nearest neighbor based aggregation algorithms for computing community scores. We show how these scores can be combined to recommend communities and test the effectiveness of the recommendations over a large real world dataset.

Categories and Subject Descriptors

H.1.2 [**Information Systems**]: User/Machine Systems—*Human information processing*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*

Keywords

Community Question routing; Group recommendation

1. INTRODUCTION

An online community is a group of users who interact with one another through Internet technologies [10]. The members of the community generally share a common interest, background, or experience and their collective goal is to contribute towards the welfare of the community members. Similar to the growth of online question answering sites, such as *Yahoo Answers*, the popularity of question-answering within communities is on a rise. E.g., *IBM* has a dedicated internal online community portal called *connections* that allows its employees to join communities, create new communities, and post questions and discussions within the communities. There are more than 150,000 communities in *connections* and question answering within these communities play an essential role in enabling information flow and collaboration among the community members. However, the overwhelming number of communities necessitates the need for a good *routing* strategy so that a new question gets routed to the appropriately focused community and thus is resolved in a reasonable time frame. This would alleviate the burden of finding the right community from the askers.

While question routing to the appropriate answerer is an important concept, and prior studies [13, 6, 7, 3] show that it enhances the user experience, a largely ignored area is question routing to the appropriate communities. Routing questions to communities instead of individual users is more useful as it can increase the likelihood of the question being answered. Also, it does not clog the bandwidth of any individual. Moreover, the collective knowledge of the community is greater that of an individual.

One of the main challenges for community question routing is the task of modeling community features. There is extensive research on modeling user behavior, but how do we aggregate that to capture community behavior? Simple aggregation techniques, such as, *min*, *max*, and *mean* do not work because they fail to take into account the fact that a community consists of a large number of knowledge seekers and only a small number of knowledge creators. A second challenge is how do we capture community norms and practices? Adherence of the question to community norms can be critical. To see this, consider an example. Let x and y be members of "java" community. Additionally, x is a member of "python" community. For a python question which only x and y can answer, greedy strategy dictates routing it to "java" community. However that question might not be acceptable to other community members. Hence community norms can be an important attribute besides user interests.

In order to address these challenges, we build upon the prior state of art in topic modeling and expertise identification and propose *knn* based algorithms to aggregate user features to compute community scores. We also propose some novel metrics, such as language analysis of questions, and inclination to respond, to model community character-

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505669.

istics like community norms. We evaluate several models that combine community scores to produce a ranked list of communities over IBM *connections* dataset. Our framework can run in real time for large scale datasets.

2. RELATED WORK

To the best of our knowledge the problem of routing a question to a focused community has not been explored previously. However, question routing to individual users has been recently explored [13, 6, 7, 3]. Zhou et al. [13] present a mechanism to find the top-k potential experts to answer a given question. They model the expertise based on users' forum postings and through the interconnections between the users. Li et al. [7] present a routing approach in a programming language forum. They use semantic information from user posts along with the post-reply relation to find the experts. Li et al. [6] exploit the hierarchical category information in forums to find potential answerers. Cao et al. [3] formulate the problem of question routing as a tree cut problem on the question graph and present a minimum description length based approach for selecting the best cut.

A closely related problem to community question routing is group recommendation [9, 1]. Group recommendation is a task of recommending items (such as games, movies) to a group of users instead of a single user. However the key difference is that for group recommendation, the recommended item must appeal to all group members. As a result, *min*, *max*, or *mean* based aggregation work well in practice for group recommendation [9], but for question routing, one must consider the fact that most of the community members are information seekers and not active contributors, so a community profile based on all its members does not work.

Another related problem is question classification which aims at putting the question into several semantic categories [11]. This is done by building a category profile based on the prior categorized questions. However typically these categories are orthogonal and they do not share topical similarities, whereas communities do not follow such constraints. In fact in most cases, communities compete with one another in the topic space. Additionally, the notion of user membership and their intra-community dynamics is not present while question categorization, but is pivotal for routing.

3. PROBLEM DEFINITION

Let $U = \{u_1, u_2, \ldots\}$ represent the set of users and $C = \{c_1, c_2, \ldots\}$ be the set of communities, such that, $u_i \in c_j$ indicates that the user u_i is a member of the community c_j. A user can be member of multiple communities.

PROBLEM 1 (COMMUNITY QUESTION ROUTING). *Given a question q asked by a user v, find a community c^*, such that the following holds:*

$$c^* = \arg\max_{c \in \mathcal{C}} \{Value\,(q, c)\} \quad (1)$$

where $Value(q, c)$ is an estimate of the quality of the answers generated by the members of the community c to question q, provided that the question is asked in community c.

In this paper, we consider the problem of finding top k communities based on their value to a given question q. The next section presents metrics that are used to model the community characteristics.

ID	Feature
$Q1$	Number of views
$Q2$	Number of unique answerers
$Q3$	Did asker answer the question
$Q4$	Word vector (normalized using Tf-Idf)
$Q5$	Topic distribution (LDA, LSA, etc.)
$Q6$	Language analysis (e.g. typos, negative/positive emotions, greetings, nouns, pronouns, etc.)

Table 1: Question features. Note that features $Q1$, $Q2$, and $Q3$ are not present for the routed question.

4. METRICS FOR QUESTIONS ROUTING

There are three different types of entities that need to be modeled for our problem. The first type is the *question* which includes the routed question and the existing questions in the communities. The second type is the *user* which includes the question asker and the community members. Finally, the third type is the *community* where the interaction between the users occur. Next, we present features extracted for these entities.

Question Features

A question typically consists of a summary title and an optional description. We combine these two fields together to create a question document (QD). Based on QD, we extract several different features as listed in Table 1. Most of these features are self-explanatory.

We compute the questions' topic distribution ($Q5$) through the Latent Dirichlet Allocation (LDA) [2] algorithm. To run LDA, we first combine all QDs within a community to create a community document CD. Then, LDA is run over CDs and once it estimates the topic-word proportions, we infer the topic distribution of an individual question. We run topic model over CDs instead of QDs directly because it leads to a more reliable estimation of topics due to two main facts: a) there are fewer communities (hundreds to thousands) in comparison to questions (tens or hundreds of thousands), and, b) CD are wordy ($\sim$7000 words) whereas QD are terse ($\sim$100 words). These factors make CD more robust to noise and thus a more reliable topic estimation.

We carried out a language analysis ($Q6$) of the question text using Linguistic Inquiry and Word count (LIWC) tool. LIWC provides scores on 80-90 features for the input text. The most prominent of those features are: a) usage of negative (and positive) words, b) usage of singular pronouns, c) usage of bad words, d) usage of greetings, and e) usage of special characters (e.g. @, #, ?, !, etc). We also computed the score for spelling mistakes through a dictionary (English words + Technical jargon) and Jaro-Winkler similarity measure. The language analysis enables us to test whether a question adheres to the community norms (e.g. censorship of some communities towards bad words, excessive personalization of a post, bad-readability of a post due to lots of spelling mistakes, sms language, or special characters).

User Features

Table 2 provides a complete list of the user features. In order to compute users' topical expertise $U4$, we consider a variant of *z-score* expertise model [12]. To estimate the

ID	Feature
$U1$	Demographics of user (job, department, location)
$U2$	Communities that user created or owns
$U3$	Communities which contain user as a member
$U4$	Topical expertise of the user
$U5$	Availability of the user
$U6$	Inclination to respond to a given user
$U7$	Language analysis of the answered questions

Table 2: User features.

ID	Feature
$C1$	Word vector of community title and description
$C2$	Topical distribution of the community
$C3$	Topical expertise of the community members
$C4$	Availability of the community members
$C5$	Inclination to respond to a user
$C6$	Language analysis

Table 3: Community features.

topical expertise, we consider topical *z-score* as follows:

$$z(t) = \frac{a(t) - q(t)}{\sqrt{a(t) + q(t)}} \tag{2}$$

where t is a topic, $a(t)$ indicates the sum of topic t's component for all the questions answered by the user, and similarly $q(t)$ represents the sum of topic t's component in all the questions asked by the user.

We estimate user availability ($U5$) by examining the most recent two weeks' hourly activity of the users and assigning a probability proportional to the frequency of activity per hour. Typically users are active for a short period of time followed by a long period of passiveness, hence examining recent two weeks gives a more accurate estimation of their current activity levels.

Inclination to respond to a user ($U6$) is computed based on the strength of the shortest path between the two users in the QA graph. The QA graph [12] is a directed weighted graph generated by connecting the question asker (x) to the answerer (y), where edge weight indicates the number of questions of x answered by y. We define the inclination to respond as the minimum weight on the shortest directed path from the asker to potential answerer. If no path exists then inclination is set to 0.5. The intuition behind this feature is that if a user u has replied to another user v in the past or if u connects to v with a high strength directed path, then u might like to reply to v in the future as well.

Community Features

Table 3 presents a list of community features. The topical distribution ($C2$) is computed using LDA over community documents (CDs). We compute the community expertise, availability, inclination, and language analysis in a similar fashion as we computed for the individual users by aggregating all the activity within the community.

4.1 Similarity Metrics

We combine the features presented in Table 1, 2, and 3 to construct several similarity metrics. These metrics are computed from the perspective of a routed question q.

Question Question Similarity Metrics

To compute the similarity of q with the existing questions in the communities, we use Kullback-Leibler divergence (KL) between the topic distributions $Q5$ as,

$$KL_{x||y} = \sum_i x_i * \log\left\{\frac{x_i}{y_i}\right\} \tag{3}$$

where x_i and y_i are the i^{th} component of the probability distribution x, y. Low KL value indicates high similarity and high KL value indicates low similarity. We use topic similarity $KL_{Q5_q||Q5_{q'}}$ to route q to the communities where similar questions were asked. However, we need to incorporate another important factor: how well those questions were answered. If those questions were not properly answered, then it might be unproductive to route q to their communities. To incorporate this factor, we propose the notion of question value $Val(p)$ which is estimated based on the number of people who viewed a question and the number of unique users who answered it.

$$Val(p) = Q1_p \cdot \log(1 + Q2_p - Q3_p) \tag{4}$$

The topic match between the routed question q and an existing question q' is then defined as:

$$q\text{-}topic(q, q') = Val(q') \cdot KL_{Q5_q||Q5_{q'}} \tag{5}$$

Similarly, we compute the match between the language features of the two questions.

$$q\text{-}lang(q, q') = Val(q') \cdot Cosine(Q6, Q6_{q'}) \tag{6}$$

where $Cosine(a, b) = \frac{a^T b}{||a||_2||b||_2}$ is the cosine similarity between two vectors a and b. Note we use cosine similarity here instead of KL because language features need not be probability distributions.

User Question Similarity Metrics

To compute user question similarity metrics, we first compute the familiarity (F) of the potential answerers (u) with the question asker (v). This factor is computed based on the hypothesis that users who share a lot of communities, have similar demographics, or have interacted with one-another previously will be more familiar with one another.

$$F(v, u) = \log\left\{\begin{array}{l} 1 + U6(v, u) + \alpha J(U1_v, U1_u) \\ \quad + \beta J(U2_v, U2_u) + \gamma J(U3_v, U3_u) \end{array}\right\} \tag{7}$$

where α, β, γ are the weight parameters and $J(a, b) = \frac{|a \cap b|}{|a \cup b|}$ is the Jaccard index between two sets a and b. We consider three metrics for finding potential answerers.

$$u\text{-}exprt(q, u) = F(v, u) \cdot KL_{Q5_q||U4_u} \tag{8}$$

where *u-exprt* estimates the topical expertise match of the user to the routed question.

$$u\text{-}lang(q, u) = F(v, u) \cdot Cosine(Q6_q, U7_u) \tag{9}$$

where *u-lang* estimates the language match between the question answered by u' previously and the routed question.

$$u\text{-}avail(q, u) = F(v, u) \cdot U5(q, u) \tag{10}$$

where *u-avail* estimates the availability of the user around the time of the post of the question.

Algorithm 1 *Global-knn*$(k, n, q, O, C, \mathcal{M})$

Require: $\mathcal{M} : q \times O \rightarrow \mathbb{R}$ is a similarity metric.
1: Sort $o \in O$ in decreasing order of their $\mathcal{M}(q, o)$. Let the sorted list be $\{o_{i_1}, o_{i_2}, \ldots\}$, s.t., $\mathcal{M}(q, o_{i_a}) \geq \mathcal{M}(q, o_{i_b})$ for all $i_a < i_b$.
2: Pick top k objects, $O_k^\star = \{o_{i_1}, \ldots, o_{i_k}\}$.
3: Compute community score $CS_G(c)$ based on the number of objects in $O_k^\star$ that belong to the community c.

$$CS_G(c) = |\{o : o \in O_k^\star \wedge o \in c\}| \quad (13)$$

4: Return top n communities based on their CS_G score.

Community Question Similarity Metrics

First, we estimate the topical match of q with community c. To do this, we use the topic distribution of the question and the community. Additionally, a community contains a brief description supplied by its owners. This description could be useful as it might hold clues to the topical interest of the community. We combine these two factors to compute topical similarity as follows:

$$c\text{-}topic(q, c) = KL_{Q5_q||C2_c} - \delta \cdot Cosine(Q4_q, C1_c) \quad (11)$$

where δ is the relative importance of community description over KL match. Now similar to *u-exprt* we compute the expertise of the community towards answering q as follows:

$$c\text{-}exprt(q, c) = KL_{Q5_q||C3_c} \quad (12)$$

In a similar fashion, we compute *c-lang* and *c-avail*.

5. RECOMMENDATION ALGORITHMS

In order to aggregate individual metrics (q-$*$, u-$*$) to compute community scores, we present two k nearest neighbor (*knn*) algorithms. The intuition behind the *knn* algorithms is that *min*, *max*, and *mean* based aggregations do not capture the true scores for a community. This is primarily due to the skew in activity levels of the community members. The two algorithms take as input the following parameters: number of recommendations n to produce, number of nearest neighbors k, a question q, a set of objects O, a set of communities C which contain the objects in O, and a similarity metric $\mathcal{M}$ between q and the objects in the set O.

Global-knn. Algorithm 1 presents *Global-knn* algorithm. It first picks k objects with the largest similarity with the routed question.[1] Then it computes the score of each community based on how many of its objects are in the top k. The n communities with the highest scores are returned.

Local-knn. Algorithm 2 presents *Local-knn* algorithm. Unlike the previous approach, it first picks top k objects per community with the largest similarity with the routed question. Then the average similarity of the top k objects constitute that community's score and the top n communities with the highest score are recommended. One issue with this algorithm is that it could be biased towards communities with a number of objects less than k. To tackle this problem, we normalize the community scores as: $\overline{CS}_L(c) = CS_L(c) * \log\{|O_k^\star(c)|\}$. Log-based normalization ensures that the communities with small number of objects are penalized by a factor of $\frac{\log k_1}{\log k}$.

[1]Note that for $*$-*topic* metrics, smaller value is preferred. This is handled by multiplying it with -1 beforehand.

Algorithm 2 *Local-knn*$(k, n, q, O, C, \mathcal{M})$

Require: $\mathcal{M} : q \times O \rightarrow \mathbb{R}$ is a similarity metric.
1: Let $O(c) = \{o : \forall o \in O \wedge o \in c\}$ be the set of objects that belong to a community c.
2: Sort objects in $O(c)$ in decreasing order of their $M(q, o)$ score. Let the sorted list be $\{o_{i_1}, o_{i_2}, \ldots\}$, s.t., $M(q, o_{i_a}) \geq M(q, o_{i_b})$ for all $i_a < i_b$.
3: Pick top k objects, $O_k^\star(c) = \{o_{i_1}, \ldots, o_{i_k}\}$ and compute the community score $CS_L(c)$ as follows:

$$CS_L(c) = \frac{1}{|O_k^\star(c)|} \sum_{o \in O_k^\star(c)} \mathcal{M}(q, o) \quad (14)$$

4: Return top n communities based on their CS_L score.

5.1 Combined Algorithms

Using the two *knn* algorithms, we get several scores for a community corresponding to each similarity metric. Let $\vec{f}_{c,q}$ represent all such scores of a community c for a routed question q. Now the problem is to use $\vec{f}$ to rank communities. To do this, we consider three mechanisms.

The first mechanism is to use *linear regression* for ranking. We first construct a binary response variable (y) which is set to 1 for the desired community and 0 for the non-desired ones. The optimal weights w_{lr} are estimated using the linear regression. The closed form solution is $w_{lr} = (F^T F)^{-1} F^T Y$, where $F = [\vec{f}_{c_1,q}\ \vec{f}_{c_2,q} \ldots]^T$ is the design matrix and $Y = [y_{c_1,q}\ y_{c_2,q} \ldots]^T$ is the output response.

The second mechanism is to generate a ranked list per score type and then merge those ranked lists. The general version of this problem is NP-hard but there are several greedy algorithms. We consider a simple yet popular *Borda count* algorithm [4]. The algorithm simply computes the aggregate rank of a community and sorts then on their aggregate rank. We consider a weighted version of the algorithm, in which weighted aggregate rank is computed. In order to learn the weights, we use an iterative reweighing scheme, where weight of one ranked list is estimated by fixing the weights of other ranked lists and so on. The weights that minimize the rank of desired communities is chosen.

The third mechanism is to cast it as a convex optimization problem with pairwise constraints. We use *ranking SVM* which is a popular algorithm for learning from ranked lists [5]. Let R_q and S_q be the set of desired and non-desired communities for question q. Then the optimal weights w_{svm} are obtained through the following minimization,

$$\begin{aligned} &minimize: && \frac{1}{2} w^T w + \lambda \sum \xi_{a,b,q} \\ &subject\ to: && \\ \forall q, \forall a \in R_q, \forall b \in S_q: && w^T(\vec{f}_{a,q} - \vec{f}_{b,q}) > 1 - \xi_{a,b,q} \\ \forall q, \forall a \in R_q, \forall b \in S_q: && \xi_{a,b,q} \geq 0 \end{aligned} \quad (15)$$

For final ranking, we sort communities in decreasing order of their $w^T \cdot \vec{f}_{c,q}$ value.

6. DATASET

We experimentally evaluate the performance of our model on IBM Connections datasets, which consists of tens of thousands of communities specific to certain goals, such as socialization, collaboration, and knowledge sharing. The breadth of topics in the communities varied from development and

Metrics	*Global-knn*			*Local-knn*		
	$P@1$	$P@5$	MRR	$P@1$	$P@5$	MRR
q-topic	0.44	0.56	0.52	0.45	0.64	0.52
q-lang	0.18	0.25	0.20	0.19	0.33	0.24
u-exprt	0.39	0.62	0.50	**0.48**	**0.66**	**0.55**
u-lang	0.13	0.32	0.21	0.24	0.41	0.31
u-avail	0.03	0.07	0.08	0.09	0.17	0.10
c-topic	–	–	–	0.37	0.54	0.43
c-exprt	–	–	–	0.29	0.46	0.34
c-lang	–	–	–	0.09	0.18	0.15
c-avail	–	–	–	0.02	0.16	0.12

Table 4: Overall performance of the two *knn* algorithms over different similarity metrics.

coding to finance and human resource to fun and outings. The community software allows community members to post questions and answers. By default the communities are marked as public, which allows non-members to read the community data. We randomly selected a set of $2,087$ public communities and crawled all their data. The crawled data consists of 59,561 questions asked by 44,318 users.

7. EVALUATION

The goal of our evaluation is two fold. First, we want to evaluate the performance of the two *knn* based aggregation algorithms with prior used aggregation techniques: *min*, *max*, and *mean*. Second, we want to compare the combined models with the individual models (e.g. based on best answerers) and explore the predictive power of different similarity metrics.

For model evaluation, we use 10 fold cross validation. In each fold, we consider 80% data for training, 10% for parameter estimation (hold out), and 10% for testing. The parameters $\alpha, \beta, \gamma, \delta$ are estimated empirically over the hold out data. For the test questions, task is to retrieve the communities where those questions were asked. Note that for each question, there is only one correct community. We also ensure that only the answered questions ($Val > 0$) are considered for training and testing.

We consider two evaluation metrics for assessing the routing algorithms. The first measure is precision at position N (P@N), which is 1 if the desired community is among the top N communities, otherwise 0.[2] The second measure is Mean Reciprocal Rank (MRR) which is the inverse of rank of the correct community averaged for all question queries.[3] These two measures are widely used in the IR domain [8].

7.1 Model + Metric Evaluation

We begin with a systematic evaluation of the similarity metrics and the two *knn* algorithms. To run these experiments we set the number of nearest neighbor to 5 ($k = 5$). We also set the number of topics to 150. Table 4 shows the performance of the *knn* algorithms. We note here that *min* and *mean* based aggregation, which are popular for group recommendation, perform badly for the routing task. The best $P@5$ amongst them is 0.55, achieved by *mean* strategy over *u-exprt*, while both the *knn* methods over the same metric perform significantly better. Additionally, we make several key observations from the table as follows:

- Similarity metrics *q-topic* and *u-exprt* perform significantly better than all other similarity metrics across all the conditions using one sided *t-test* ($p \sim 0$). Among those two metrics, *u-exprt* is significantly better.
- *Local-knn* is significantly better than *Global-knn* for all similarity metrics using one sided *t-test* ($p \leq 0.01$), indicating that computation of community scores based on top users/questions with in a community is more effective than first picking top users/questions globally and then computing the community scores through a voting mechanism.
- Similarity metrics *c-*$*$, which are computed by aggregating all activity within a community naively without considering semantics of users and questions, performs significantly worse than their *q-*$*$ and *u-*$*$ counterparts across all conditions using one sided *t-test* ($p \sim 0$).

Fig. 1a plots $P@N$ for *u-*$*$ and *q-*$*$ similarity metrics. We observe an increase in precision by 70–150% from $N = 1$ to $N = 20$. The precision tapers off for large N. We also note that if a metric is better that another metric, it is consistently better for all N. This consistency indicates high reliability of these metrics.

7.1.1 Topic + Similarity Function Evaluation

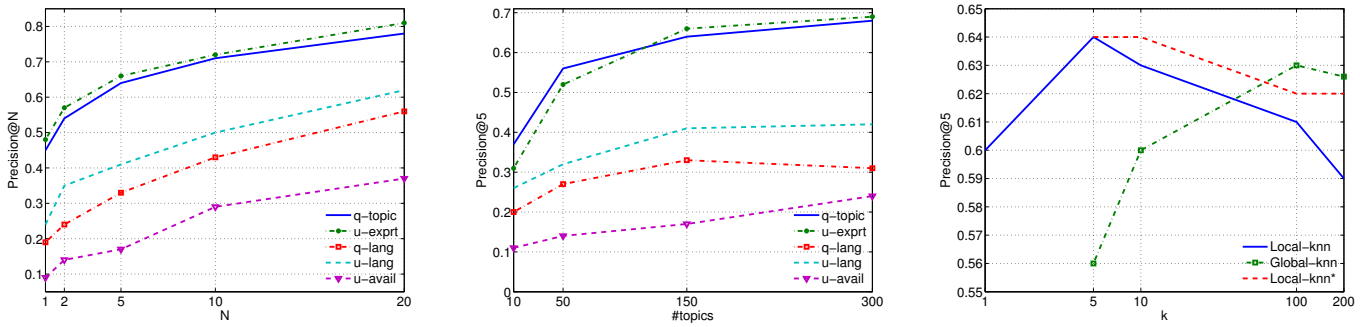
Here we evaluate the effectiveness of the topic model towards model performance. Note that the first step in computation of the similarity metrics is the inference of the topic distribution of the questions. One of the key parameters required by LDA topic model is the number of topics. Fig. 1b shows the precision of *Local-knn* algorithm for several choices of number of topics (#topics). It shows that an increase in #topics leads to an increase in the performance. This increase is large for small values (around 50%) and is small for large values (around 1-2%). Intuitively, a small value of #topics leads to under-fitting in the topic space, which in turn leads to increase in false positives. On the other hand, a large value of #topics could lead to over-fitting, which in turn might eliminate desirable communities from the candidate set. Additionally, a large #topics leads to more iterations of the LDA algorithm to converge. Overall this result shows that the models can be sensitive to the choice of number of topics.

7.1.2 Number of Nearest Neighbors

An important parameter required by the *knn* algorithms is the number of neighbors k. Fig. 1c shows their performance for different values of k. We note that $k = 1$ leads to a *max* based aggregation and $k = \infty$ is *mean* based aggregation. Both these aggregation appear to be significantly worse in comparison to $k = 5$, highlighting that trivial aggregation mechanisms are ineffective.

We observe that the performance of *Global-knn* increases and then ultimately decrease. The increase makes sense because as k is increased the score of a topically relevant community with a lot of similar questions would increase. However as k becomes large the performance degrades as the algorithms get biased towards communities with a lot of questions irrespective of their similarity.

[2]P@N is a variant of precision. It is more suitable here because there is only one correct community per question.
[3]Since there is only one correct community per question, MRR is more suitable than Mean Average Precision.



(a) Performance of the *Local-knn* algorithm for different values of N.

(b) Performance of the *Local-knn* algorithm for different choices of number of topics.

(c) Performance of the *knn* algorithms over *q-topic*. *Local-knn∗* represents *Local-knn* algorithm with log scaling.

Figure 1: Performance of different models.

Dataset	Model	*P*@1	*P*@5	*MRR*
Enterprise	*Linear regression*	0.56	0.77	0.66
	Borda count	0.57	0.80	0.67
	Ranking SVM	**0.62**	**0.81**	**0.71**
	B1: random	0	0.003	0.004
	B2: size	0.08	0.23	0.16

Table 5: Performance of the recommendation models. For *ranking SVM*, we set the parameter $\lambda = 1$.

We observe a similar trend for the *Local-knn* algorithm. However here the performance starts to drop for $k > 5$. This is because as k is increased the algorithm gets biased towards communities with fewer questions. To solve this problem, we multiply the community scores with $\log(\min\{k, \#\text{object}\})$. Fig. 1c (*Local-knn∗*) shows that the performance of log scaling remains steady unlike its unscaled counterpart.

7.2 Combined Model Evaluation

Here we test the performance of the three combined models and two baseline models (B1 and B2). The B1 model generates a random permutation of communities. The B2 model sorts communities in order of their decreasing sizes, so it is sensitive towards communities with lots of objects. Table 5 shows the performance of the models. We observe that the combined models perform significantly better than the models which route a question to the community of best individual to answer that question (Table 4, *u-exprt*), using one sided *t-test* ($p \sim 0$). They also perform significantly better than the baselines, and the individual models (Table 4). Among the combined models, the *ranking SVM* performs significantly better, using one sided *t-test* ($p \leq 0.01$). Intuitively this makes sense, because the pair-wise constraints lead to a direct optimization of *P*@1 accuracy measure.

7.2.1 Relative Importance of Similarity Metrics

Here we estimate the relative predictive power of different community scores through $r2$ of linear regression. Expertise and topic similarities lead to an $r2$ of 0.43. Addition of language features leads to a jump in $r2$ by 10%. On a deeper analysis, we found that the language features, such as usage of technical terms, personalization, friendliness (hello, thanks, nice, please), negative sentiments are the most useful language features.

8. CONCLUSION

In this paper, we address a novel problem of question routing to user communities. We used several prior proposed measures and added novel measures for modeling different entities. We performed a comprehensive evaluation of our proposed framework and showed its effectiveness over a large real world dataset. Our work is a first step to address the community question routing problem and as part of our future work, we will explore different ranking algorithms and test our framework on several other datasets.

9. REFERENCES

[1] S. Amer-Yahia, S. B. Roy, A. Chawlat, G. Das, and C. Yu. Group recommendation: semantics and efficiency. *Proc. VLDB Endow.*, 2(1):754–765, Aug. 2009.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. In *NIPS*, pages 601–608, 2001.

[3] Y. Cao, H. Duan, C. Lin, Y. Yu, and H. Hon. Recommending questions using the mdl-based tree cut model. In *WWW*, pages 81–90. ACM, 2008.

[4] D. Coppersmith, L. Fleischer, and A. Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *SODA*, pages 776–782. ACM, 2006.

[5] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[6] B. Li, I. King, and M. Lyu. Question routing in community question answering: putting category in its place. In *CIKM*, pages 2041–2044. ACM, 2011.

[7] W. Li, C. Zhang, and S. Hu. G-finder: routing programming questions closer to the experts. In *ACM Sigplan Notices*, volume 45, pages 62–73. ACM, 2010.

[8] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[9] M. O'Connor, D. Cosley, J. A. Konstan, and J. Riedl. Polylens: A recommender system for groups of user. In *ECSCW*, pages 199–218, 2001.

[10] J. Preece and D. M. Krichmar. Online communities: Design, theory and practice. *Journal of Computer Mediated Communication*, 10(4), 2005.

[11] D. Zhang and W. S. Lee. Question classification using support vector machines. In *SIGIR*, pages 26–32, 2003.

[12] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. In *WWW*, pages 221–230. ACM, 2007.

[13] Y. Zhou, G. Cong, B. Cui, C. Jensen, and J. Yao. Routing questions to the right users in online communities. In *ICDE*, pages 700–711. IEEE, 2009.