2012 32nd IEEE International Conference on Distributed Computing Systems

SOS: A Distributed Mobile Q&A System Based on Social Networks

Ze Li and Haiying Shen and Guoxin Liu
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29631
Email: {zel, shenh, guoxinl}@clemson.edu

Jin Li
Microsoft Research
Redmond, WA 98052
Email: jinl@microsoft.com

***Abstract*—Recently, emerging research efforts have been focused on question and answer (Q&A) systems based on social networks. The social-based Q&A systems can answer non-factual questions, which cannot be easily resolved by web search engines. These systems either rely on a centralized server for identifying friends based on social information or broadcast a user's questions to all of its friends. Mobile Q&A systems, where mobile nodes access the Q&A systems through Internet, are very promising considering a rapid increase of mobile users and the convenience of practical use. However, such systems cannot directly use the previous centralized methods or broadcasting methods, which generate high cost of mobile Internet access, node overload, and high server bandwidth cost with the tremendous number of mobile users. We propose a distributed Social-based mObile Q&A System (SOS) with low overhead and system cost as well as quick response to question askers. SOS enables mobile users to forward questions to potential answerers in their friend lists in a decentralized manner for a number of hops and then resort to the server. It leverages lightweight knowledge engineering techniques to accurately identify friends who are able to and willing to answer questions, thus reducing the search and computation costs of mobile nodes. The trace-driven simulation results show that SOS can achieve a high query precision and recall rate, a short response latency and low overhead. We have also deployed a pilot version of SOS for use in a small group in Clemson University. The feedback from the users shows that SOS can provide high-quality answers.**

I. Introduction

Traditional search engines such as Google [1] and Bing [2] have been significantly impacting our everyday lives in information retrieval. To improve the performance of search engines, social search engines [3–10] have been proposed to determine the results searched by keywords that are more relevant to the searchers. These social search engines group people with similar interests and refer to the historical selected results of a person's group members to decide the relevant results for the person.

Although the search engines perform well in answering factual queries for information already in a database, they are not suitable for non-factual queries that are more subjective, relative and multi-dimensional (e.g., can anyone recommend a professor in advising research on social-based Q&A systems?), especially when the information is not in the database (e.g., suggestions, recommendations, advices). One method to solve this problem is to forward the non-factual queries to humans, which are the most "intelligent machines" that are capable of parsing, interpreting and answering the queries, provided they are familiar with the queries. Accordingly, a number of expertise location systems [11–14] have been proposed to search experts in social networks or Internet aided by a centralized search engine. Also, web Q&A sites such as Yahoo!Answers [15] and Ask.com [16] provide high-quality answers [17] and have been increasingly popular.

Recently, emerging research efforts have been focused on social network based question and answer (Q&A) systems [17–22], in which users post and answer questions through social network maintained in a centralized server. As the answerers in the social network know the backgrounds and preference of the askers, they are willing and able to provide more tailored and personalized answers to the askers, enhancing the satisfaction on the Q&A sites. The social-based Q&A systems can be classified into two categories: broadcasting-based [17–19] and centralized [20–22]. The broadcasting-based systems broadcast the questions of a user to all of the user's friends. The centralized systems [20–22] rely on a centralized server to identify possible answerers to a question without broadcasting. The centralized server constructs and maintains a social network of users, and searches the answerers for a given question from the asker's friends, friends of friends and so on.

In respect to the client side, the rapid prevalence of smartphones has boosted mobile Internet access, which makes the mobile Q&A system as a very promising application. The number of mobile users who access Twitter [23] increased 182% from 14.28 million in Jan 2010 to 26 million in Jan 2011. It was estimated that Internet browser-equipped phones will surpass 1.82 billion units by 2013, eclipsing the total of 1.78 billion PCs by then [24]. The mobile Q&A systems enable users to ask and answer questions anytime and anywhere at their fingertips. However, the previous broadcasting and centralized methods are not suitable to the mobile environment, where each mobile node has limited resources. Broadcasting questions to all friends of a user generates a high overhead to the friends, since many friends (including those unlikely to answer questions) receive questions. Also, broadcasting results in many costly interruptions to users by sending questions that they cannot answer and increase their workload of looking for questions that they can answer through a pool of received questions. The centralized methods,

1063-6927/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDCS.2012.44

by serving a social network consisting of hundreds of millions of mobile users (which are also rapidly increasing), suffer from high cost of mobile Internet access, high query congestion, and high server bandwidth and maintenance costs. It was reported that Facebook spent more than 15 million per year for server bandwidth costs and data center rent in addition to 100 million for purchasing 50,000 servers to release the high burden of traffic [25].

To tackle the problems in the previous social-based Q&A systems and realize a mobile Q&A system, a key hurdle to overcome is: *How can a node identify friends most likely to answer questions in a distributed fashion?* To solve this problem, in this paper, we propose a distributed Social-based mObile Q&A System (SOS) with low node overhead and system cost as well as quick response to question askers. SOS is novel in that it achieves lightweight distributed answerer search, while still enabling a node to accurately identify its friends that can answer a question. We have also deployed a pilot version of SOS for use in a small group in Clemson University[1]. The analytical results of the data from the real application show the highly satisfying Q&A service and high performance of SOS.

SOS leverages the lightweight knowledge engineering techniques to transform users' social information and closeness, as well as questions to IDs, respectively, so that a node can locally and accurately identify its friends capable of answering a given question by mapping the question's ID with the social IDs. The node then forwards the question to the identified friends in a decentralized manner. After receiving a question, the users can decide to forward the question or answer the questions if they can. The question is forwarded along friend social links for a number of hops, and then resorts to the server. The cornerstone of SOS is that a person usually issues a question that is closely related to his/her social life. As people sharing similar interests are likely to be clustered in their social network [26], the social network can be regarded as social interest clusters intersecting with each other. By locally choosing the most potential answerers in a node's friend list, the queries can be finally forwarded to social clusters that have answers for the question. As the answerers are socially close to the askers, they are more willing to answer the questions compared to strangers in the Q&A websites. In addition, their answers are also more personalized, trustable and accurate.

SOS is featured by three advantages:

(1) *Decentralized.* Rather than relying on a centralized server, each node identifies the potential answerers from its friends, thus avoiding the query congestion and high server bandwidth and maintenance cost problem.

(2) *Low cost.* Rather than broadcasting a question to all of its friends, an asker identifies the potential answerers who are very likely to answer this question, thus reducing the node overhead, traffic and mobile Internet access.

(3) *Quick response.* Due to the close social relationship between the question receivers and an asker, the question receivers are likely to be willing to provide answers quickly.

[1]The demo of the application and call for participation can be found from `http://people.clemson.edu/~shenh/`.

The contributions are summarized as follows:

(1) As far as we know, it is the first work to design a distributed Q&A mobile system based on social networks, which can be extended to low-end mobile devices. The system can tackle the formidable challenge facing distributed systems: precise answerer identification.

(2) We propose a method that leverages lightweight knowledge engineering techniques for accurate answerer identification.

(3) We propose a method that considers social closeness in addition to interest similarity in question forwarder selection in order to increase the likelihood of the receiver to answer/forward the question.

(4) We have conducted extensive trace-driven simulations based on the crawled data from Yahoo!Answer and Twitter with regards to node interactions in online Q&A systems and online social networks. Experimental results show the high answerer identification accuracy, low cost and short response delay of SOS.

(5) We have deployed a pilot version of SOS for use in a small group in Clemson University and revealed interesting findings in the mobile social-based Q&A system. Though Google earns a little higher user satisfaction degree than SOS on factual questions, users have much higher satisfaction degree on SOS for non-factual questions than Google. Also, socially close users tend to respond questions quickly.

Note that we do not endorse a complete removal of the centralized server from the system. We believe that dedicated servers still play an important role in the system, particularly when a node cannot find answerers in the social network. The rest of the paper is organized as follows. Section II presents related work. Section III present the design of SOS. Section IV and Section V show the trace-driven simulation results and real testbed results. We conclude this paper with remarks on future work in Section VI.

II. Related Work

While there has been relatively little research on distributed Q&A systems based on social networks, we take a slightly larger view of the problem space and compare SOS with social search, expertise location, and online Q&A systems.

Social search: In order to improve the user experience in a web search engine [1, 2], a number of works have been proposed to enable users to find resources by using social annotations or bookmarks. Evan *et al.* [3] pointed out that social interactions play an important role throughout the search process, and suggested that sharing search information among people may be valuable to individual searchers. The Phoaks [4], Answer Garden [5] and Designer Assistant [6] social search systems attempted to enable social interactions when existing information spaces are inadequate in providing experts' contact information. Amitay *et al.* [7] assumed that the interests of a searcher's friends provide a good prediction for the searcher's preferences, David *et al.* [8] proposed to re-rank the searched results by considering the strength of the

relationship between the results and the searchers. Kolay *et al.* [9] studied how social bookmarked URLs lead to new or high-quality content on the Web. Bao *et al.* [10] proposed a SocialSimRank algorithm to calculate the similarity between social annotations and web pages as well as a SocialPageRank algorithm to capture the popularity of web pages. However, the social search aims to improve the web search engine, which perform poorly in non-factual questions [18].

Expertise location: Chen *et al.* [11] proposed an open system to recommend potential research collaborators for scholars and scientists based on the structure of the coauthor network and a user's research interests. Lin *et al.* [12] introduced SmallBlue, which is a social network search engine used to help IBM employees find and access expertise and information through their own social networks. ReferralWeb [13] mined public Web documents for the knowledge about potential experts through webpage content analysis. Expertise Recommender [14] studied software source control systems and technical support databases in order to find expertise. However, these systems only try to identify experts, but do not have mechanisms to ensure that the identified experts are willing to help.

Online Q&A systems: Numerous online Q&A systems exist in the Internet [15, 16], in which anonymous users can post questions and respond to others' questions. However, the systems cannot guarantee quick response of posted questions. Morris and Teevan [18, 19] studied how people use status message in a social network to ask questions. By posting questions on his/her status wall, a user can broadcast the questions to all of his/her friends. Hsieh *et al.* [17] proposed a market-based Q&A service called MiMir, in which all questions are broadcasted to all users in the system. However, broadcasting a user's question to all of his/her friends only enables direct friends to see the question, generates high cost and produces interruptions to friends who are unable to answer the question. White and Richardson [20, 21] developed a synchronous Q&A system called IM-an-Expert, which automatically identifies experts via information retrieval techniques and facilitates real-time dialog via instant messaging without broadcasting. However, it also focuses on the direct friends of a user, and the synchronous communication faces challenges such as interruption costs and the availability of friends during the questioning time. Aardvark [22] is a centralized Q&A system, in which the centralized server receives a user's question, identifies and forwards the question to the most appropriate person in the Aardvark community. However, the centralized system structure may suffer from high query congestion, high server bandwidth and maintenance costs. As far as we know, SOS is the first distributed Q&A system that enables nodes to forward queries to efficiently and accurately find answerers.

III. System Design

A. Question Routing

SOS incorporates an online social network, where nodes connect each other by their social links. As shown in Figure 1, a registration server is responsible for node registration. Each user has an interest ID, which represents his/her interest.

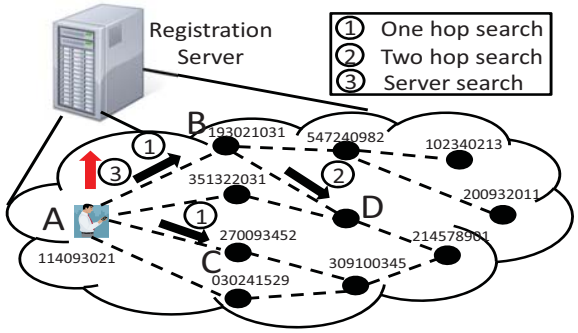


Fig. 1: Querying process in SOS.

The closeness of two user's interest IDs means the similarity between the two users' interests. Users sharing more common interests with an asker are more likely to be able to answer the asker's questions. Also, users having shorter social distances with an asker are more likely to be willing to answer the asker's questions.

SOS has a metric *similarity* (S) that measures the likelihood of a node to be able and willing to answer another node's question. It is determined by the interest similarity between the question's interest and the receiver's interest as well as the social closeness between the question receiver and sender. SOS defines a constant K, which is the largest number of friends that a node can send/forward a question in its friend list. SOS allows each node to define TTL, which is the maximal number of hops that a question can be forwarded. A node determines TTL depending on how urgent the question is. Figure 1 shows the question routing process in SOS. After asker A initiates a question, it forwards the question to the top K friends (nodes B and C) who have the highest S in its friend list with the question. A question receiver replies to A if it has an answer for the question. Otherwise, the user forwards the question to its top K friends in its own friend list in the same manner (B to D) and reduces TTL by 1. The question is forwarded along node social links until TTL=0. If the question initiator has not received an answer after delay above its specified threshold corresponding to TTL (e.g., 1 hour), it sends the question to the server that holds a discussion board, which can be accessed by all users in the system. The discussion board serves as a store for unsolved questions in the distributed system. Then, the questions in the discussion board are handled as in online Q&A systems. From this process, we can see that three problems need to be resolved.

- How to derive the interests of a question or a user (Section III-B)?
- How to infer the interest from a question and a user for more accurate answerer identification (Section III-C)? For example, from "Tom is a male CS student who likes reading book," we can infer "Tom likes fiction" so that he can be identified as the answerer for the question "who is the author of star war?" Without inference, Tom may not be identified as an answerer for the question.
- How to locate K friends in the friend list by considering both interest similarity and social closeness (Section III-D)?

Below, we introduce the solutions for these three problems.

B. Question/User Interest Representation

When a user first uses the SOS system, s(he) is required to complete his/her social profile such as interests, professional background and so on. Based on the information, the registration server recommends friends to the user, and the user then adds friends into his/her friend list. The friend lists along with the profiles of the friends are stored in the local database of the user as shown in Figure 2. Users A, B and C connect with each other based on their social relationships, and each user has a social profile. Each node maintains the social identify representation (social ID in short) of each of its friends, which is used to measure the capability and willingness of a friend to answer the node's question. The social ID of a friend is retrieved by preprocessing the social information of the friend. As shown on the right part of Figure 3, to preprocess a friend based on his/her social information, the node first derives the first-order logic representation (FOL) [27], then conducts first-order logic inference to infer the friend's interests, from which it retrieves interest ID. The node then combines the interest ID with the social closeness between itself and the friend to calculate the friend's social ID (we will explain the combination in Section III-D) represented by a numerical string (e.g., 3202001001).

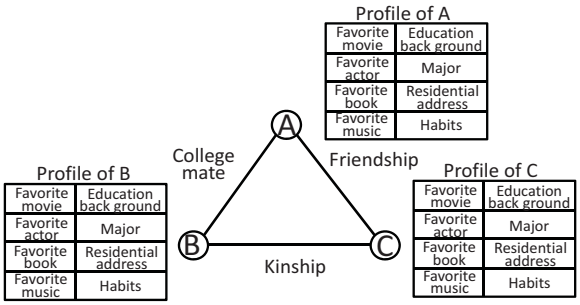


Fig. 2: An example of a node's social network.

Figure 3 shows the local answerer selection process for forwarding a question in one mobile node in the SOS system. To parse a question, the node first processes the questions in the nature language, it then represents the question in the FOL format and uses the FOL inference to infer the question's interests. Finally, it transforms the question to a numerical string (question ID). After the node parses its initiated question to an ID, it then finds the top K friends whose social ID are closest to the question's ID. Subsequently, it forwards the question to the identified friends.

For instance, an asker may ask a question "Where is the best place to watch the movie Avatar in Clemson?". The corresponding keyword list of this question is resolved to the FOL format [where, place, movie, Avatar, Clemson] after natural language processing. After the FOL inference, the FOL format is changed to [movie(sci-fi), director (James Cameron), place(Clemson)], which is later encoded as a numerical string such as 3200001000. Similarly, a student in Clemson University who likes to watch sci-fi movie is represented as [movie(sci-fi), career(student), place(clemson)] after the FOL inference and be further encoded as 3202001001. Because the

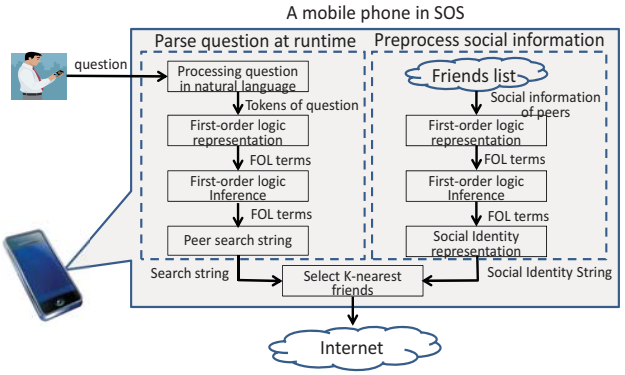


Fig. 3: Answerer selection process for forwarding a question in one node.

student's social ID is close to the question's ID, he is identified as one of the K top friends to send the question to. By comparing the similarity between a question's ID and its friend's social ID, a node can identify its friends that are willing and able to answer/forward questions. More details of the parsing process for a question or for a user is demonstrated in Figure 4 and Figure 5, respectively. The figures list the three steps in the process: FOL representation, FOL inference, and ID transformation. Below, we introduce the details of the three steps.

1) Preliminary of the first-order logic (FOL): FOL is a powerful tool to describe objects and their relationships in real life. In FOL, the users need to define basic rules or axioms, which serve as the base of the inference. For example, the FOL for an axiom in nature language "All computer science (CS) male students who like reading like sci-fi movies" is

$$(\forall x, y)(\text{CS(x)} \wedge \text{male(x)} \wedge \text{Activity(Reading)} \Rightarrow \text{like(y)}),$$

where "CS(x)", "male(x)", "Activity(Reading)", and "like(Sci-Fi)" are *predicate symbols*, and $\wedge$ is *connectives symbol*. In a FOL representation, connectives symbols (e.g., $\sim$, $\wedge$) and *quantifiers* (Universal(8) and Existential (9)) logically connect *constant symbols*, *predicate symbols* which map from individuals to truth values (e.g., green (Grass)) and *function symbols* which map individuals to individuals (e.g., father-of(Mary)=John). These symbols represent *objects* (e.g., people, houses, numbers), *relations* (e.g, red, is inside) and *functions* (e.g., father of, best friend), respectively.

2) First-order logic representation: A question or user profile information is always expressed in the natural language. To convert a question or profile information into a format that a computer can understand, we can use part-of-speech tagging [28] or Modern natural language processing (NLP) techniques [29] to divide the question into a group of related words expressed by words, 2-word phrases, the wh-type (e.g., "what", "where" or "when"). Then we transform questions into the FOL representation. First, we parse the natural language into token keywords. These token keywords will be the constant symbols in the FOL representations. The step 1 in Figure 4 shows an example of FOL representation of the query. The keywords of the question "Where is the best cinema in location A?" is "cinema" and "location A".

We also transform user social information into the FOL representation. As shown in Figure 2, a user's social infor-

mation includes his/her profile (e.g., job, hobby, favorites) and the social relationships with other users (e.g., kinship, colleague, classmate). A user's local database stores his/her own profiles, social relationship to and the profile of each of his/her friends. Specifically, a node first represents its profile in the form of *name-values pairs* such as "movies: Avatar, The Social Network", "music: Hey, Jude". That is, each interest is indexed by a unique name (e.g., movie, music), and the interest can have several values. The syntax *name(value)* is then transformed to the FOL representation expressed by predicate symbols. For example, the FOL representation of the previous example is "movie(Avatar)","movie(The Social Network)", "music(Hey, Jude)". The first step in Figure 5 shows an example of FOL representation of a node's profile. Tom has several profile information in *name-value pair* format. He has favorite book A_1, A_2, A_3. This information is transformed into FOL representation $Fa\_bk(A_1)$, $Fa\_bk(A_2)$ and $Fa\_bk(A_3)$.

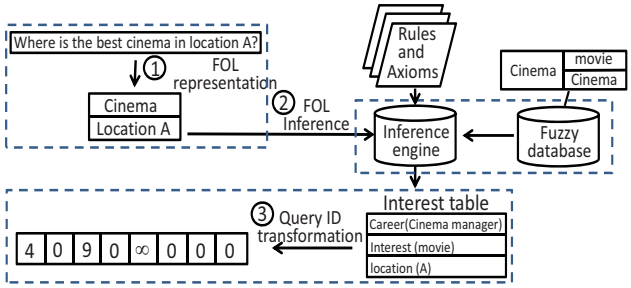


Fig. 4: An example of FOL inference for a question.

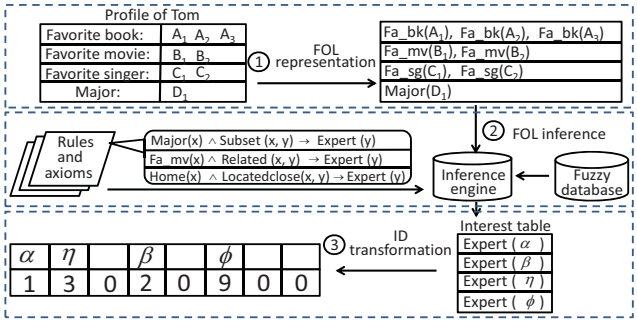


Fig. 5: An example of FOL inference for a user's social information.

C. First-order Logic Inference

As shown in the step 2 in Figure 4 and Figure 5, the FOL inference component consists of three parts: (1) fuzzy database, (2) rules and axioms, (3) inference engine. The goal of the inference is to identify node interests represented by a numerical string that can accurately represent the capability of a node to answer questions. The fuzzy database is used to store words that have relationships, including subset, alias, related, with the information in profiles. For example, Related(cinema)=movie, Subset(computer science, algorithm), Alias(USA)=US.

The rules and axioms provide basic formulas for the inference. For example, given a rule "Major(x)∧ Subset(x)=y ⇒ expert(y)", we search Major(x_1) in a person's profile and search Subset(x_1)=y_1 in the fuzzy logic database. If both of the entries can be found, then we can infer that the people should be an expert of y_1. If Tom majors in computer science, and Algorithm is a subset of computer science, then Tom should be good at answering questions on Algorithm.

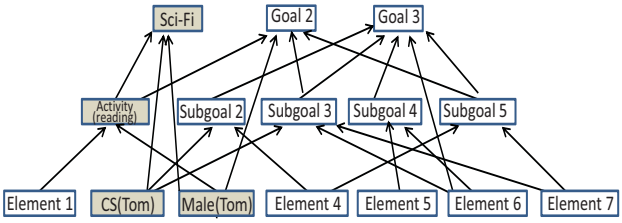


Fig. 6: Lattice in an inference engine.

Inference engine is the place that the elements and the rules are evaluated. The inference engine sets each interest category as an inference goal and builds lattice inference structure, as shown in Figure 6, to connect all the FOL symbols with the goals. Each node in the lattice is a FOL syntax symbols and the arrows represent the connective symbols that connect the symbols. By mapping the syntax symbols shown in the question (or social information) and fuzzy database, we can trace up from the basic elements to the final goal. For example, as shown by the gray box in Figure 6, if three syntax symbols CS(Tom), Male(Tom) and Activity(Reading) are all satisfied, we can infer that the goal Sci-Fi is satisfied for Tom, i.e., Tom can answer the question about Sci-Fi. We can see from Figure 4 and Figure 5 that after the elements pass the inference engine, the previous FOL representation is transformed into the interest table listing the interests of the question (or user). As shown in the step 3 of Figure 4 and Figure 5, the interests of a question (or user) are transformed to a numerical string to represent the knowledge field of a question (or a user).

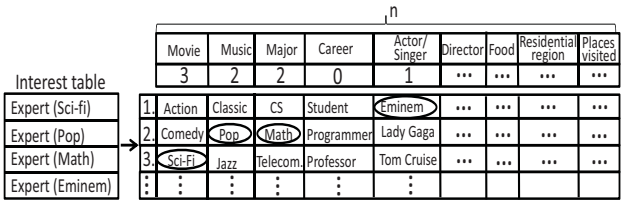


Fig. 7: An example of interest arrays.

Next, we introduce how to generate such a numerical string. Below, interest and category are interchangeable used, and a category interest means an element in the category. Figure 7 shows an example of category interest arrays. The top line lists all categories in the system. The category interests of a category are in the column below the category. These interests are alphabetically sorted. Each interest in a category uses its entry position to represent itself. Each category string for a question (or user) has n digits (e.g., 322023150). Each digit represents a category interest. For example, string 322023150 denotes "Sci-Fi, Pop, Math, Lady Gaga..." From the categories of a question (or user) in the category table, if the profile information of a person matches an element in a category, the index of the element is the digit in the corresponding

position of the numeric string that represents the category. If a person does not have any element in a category, the category's position in numeric string is set to 0. If the person can match any specific element in this category, the category is set to ∞. If the person have several elements for a category, we use "-" to concatenate the index of the category. For example, string 1-2-322023150 denotes "Action—Comedy—Sci-Fi, Pop, Math, Lady gaga..." using categories as shown in Figure 7. The users periodically update metadata containing their questions and answering statistics to the registration server. Based on these metadata, the rules and axioms can be added and updated to reflect the most current behaviors of the users in the system. These updated rules and axioms are then remotely configured in the mobile phones of the users. From this design, we can see that if two sets of inferred interests (from questions or users) are similar to each other, they will have similar (question or interest) IDs.

D. Similarity Value Calculation.

After users' social information and questions are transformed into numerical strings, the similarity between a user and a question can be calculated based on two parts: interest similarity between the user and question, and social closeness between the question sender and receiver.

Interest similarity: To evaluate the interest similarity of a question of user i and a user j, we use a method proposed in [30]. We use Q_i and I_j to denote the interest strings of the question of user i and the user j respectively. We use n to denote the number of interest categories of interest elements owned by Q_i but not by I_j; use l to denote the number of categories of interest elements owned both by Q_i and I_j, and m the number of categories of interest elements owned by I_j but not by Q_i. Then the similarity of two strings is defined as:

$$\text{sim}_I(i,j) = \frac{l+1}{2}\left(\frac{1}{l+n+2} + \frac{1}{l+m+2}\right) \quad (1)$$

The value of $sim()$ ranges in the classical spectrum $[0, 1]$, and it represents the level of likelihood that two strings under comparison are actually similar. The complete overlapping of the two string $(n = m = 0)$ tends to the limit of 1 as long as the number of common features grows. The underlying idea of Equation (1) is that two strings with longer complete overlapping should have higher similarity than the two strings with less complete overlapping. In the case of no overlapping $(l = 0)$, the function approaches to 0 as long as the number of non-shared entries grow. It indicates that for two strings with larger number of entries, if they share no common entries, it is more likely that they have smaller similarity than the string with smaller number of entries and share no common entries.

Social closeness: The social closeness directly affects the willingness of people to answer or forward questions. Several recent works have studied how to effectively calculate the social closeness between two users [31, 32]. However, to reduce the load on the mobile devices, SOS directly lets user rate each friend with a closeness value, which is represented by $sim_C(i, j)$. Therefore, the final similarity between a user i's question and its friend j can be calculated as

$$sim_{(i,j)} = \chi sim_I(i,j) + (1-\chi) sim_C(i,j), \quad (2)$$

where χ is a parameter that satisfies $\chi \in [0, 1)$, which is used to adjust the weight of the social closeness $sim_C(i, j)$ and interest similarity. We confine the number of search hops to 3 hops since the social trust between two nodes decrease exponentially with distance. This relationship has been confirmed by other studies [33, 34]. Binzel *et al.* [33] discovered that a reduction in social distance between two persons significantly increases the trust between them. Swamynathan *et al.* [34] found that people normally conduct e-commerce business with people within 2-3 hops in their social network.

Algorithm 1 shows the pseudocode of the friend selection algorithm for each node. As shown in Line1 - Line8, similarity calculation has a time complexity of O(n). Line10 - Line14 show a K-node selection algorithm with a time complexity of O(f), where f is the number of a node's friends. Therefore, a node can selects K nodes with the highest similarity value with the question from its friend list in linear time. As $sim_C(i, j)$ can be pre-processed, only $sim_I(i, j)$ needs to be calculated at run time. As the number of keywords in a question is generally very small, the calculation of $sim_I(i, j)$ costs few computation resources of the mobile devices. Therefore, the friend selection algorithm has very low complexity.

Algorithm 1 Pseudocode of the friend selection algorithm conducted by node n_i.

```
1: if Received a question && Cannot answer query then
2:    for All nodes n_j in the FriendList do
3:       Calculate sim_(i,j)
4:    end for
5:    K-node-list ← Select_K_highest_node().
6:    Send the question to the K nodes in K-node-list
7: end if
8:
9: List Select_K_highest_node() {
10:   find the Kth largest element
11: ← QuickSort partition around the Kth largest element
12: return First-K-Node-List
13: }
```

IV. Performance Evaluation

In order to simulate the features of people interactions in online Q&A sites and social networks, we crawled about 9419 questions posted in the "Entertainment & Music–Movies" section in Yahoo!Answer, and 2559 tweets posted by a user with username ReadWriteWeb [35] and his/her followers in Twitter between Oct. 5, 2010 and Oct. 26, 2010. The questions that are used to evaluate the SOS system are from Yahoo!Answers. Since Yahoo!Answers does not have user profile information, we crawled 1000 users from Facebook to form a social network. We used one user as a seed and used breadth-first search to crawl their personal profile information. We ignored users that did not fill out their profiles. The crawling stopped when 1000 users were crawled. Users' profiles contain their current locations, education backgrounds, hobbies and interests, such as books, movies, music and television programs. This information was parsed and conversed to FOL and finally encoded

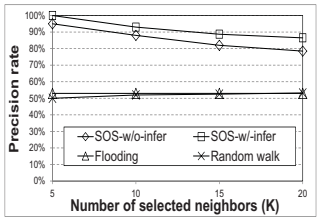


Fig. 8: Query precision rate vs. K.

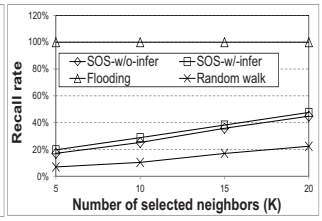


Fig. 9: Query recall rate vs. K.

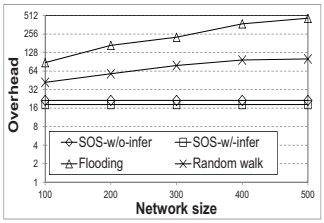


Fig. 10: Communication overhead.

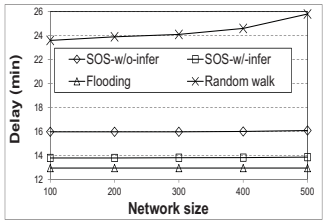


Fig. 11: Communication delay.

as strings using the method introduced previously. In the experiment, we focused on evaluating the questions related to movies, because most of the Facebook users filled out a large amount of information in the movie section in their profiles.

Since the Facebook data set is separated from the Yahoo!Answers data set, we cannot directly tell which persons can answer which questions. Therefore, to make the experiment operable, we focused on the questions in the Movie categories in Yahoo!Answers, where we selected 100 questions having keywords that can be mapped to a Facebook user's profile interests. For each of the 100 questions, we assigned the answers of the question to the Facebook users whose profile interests match most to the question's keywords; one answer to one Facebook user. The 100 questions were randomly assigned to the Facebook users to ask. We set χ to 0.5 to balance the impact of interest similarity and social closeness. In order to build the correlations between actors, movie companies, directors, we imported movie data in Internet Movie Data Base (IMDB) into the fuzzy database. Thus, when a question is issued about a certain actor A, we can search the movies, directors that are related to actor A. Then, the people who are fond of the movies or the directors that related to the actor are considered as the potential questions answerers with the inference engine.

We use RLA to denote the number of ReLevant Answers to a question existing in the system (i.e., the number of answers to a question in Yahoo!Answers), and RTA to represent the number of ReTrieved Answers in the SOS system for a question. By default, the number of friends K that a user selects for forwarding the question was set to 5. The number of hops in a social network (social hops in short) that a question is forwarded was set to 3.

In the experiments, we focus on the following metrics.

- *Precision rate:* it is a measure of exactness of the returned results: Precision $= (|\text{RLA} \bigcap \text{RTA}|)/|\text{RTA}|$
- *Recall rate:* it is a measure of completeness of the returned results: Recall $= |\text{RLA} \bigcap \text{RTA}|/|\text{RLA}|$
- *Overhead:* it is the number of messages transmitted in the system during the entire simulation process.
- *Delay:* it is the time duration between a query is issued and the first answer is received.

In our evaluation, the users returned by the different approaches are judged correct if the user has the correct answer to the question as shown in the Q&A data set.

We compare the routing performance of SOS with Flooding routing and Random walk routing. The Flooding method was mainly proposed for online social networks, in which a question is flooded to all nodes in the network. The Random walk method can mimic the question/answer behavior pattern of a user in the online Q&A sites, in which the question is randomly visited by different users until receiving an answer. In the experiments of Random walk, each node randomly sends a question to $L = K$ randomly selected friends until the answer is found. The number of search hops in SOS is limited to three. We use SOS-w/-infer to denote SOS with the FOL inference engine and SOS-w/o-infer to denote SOS without the FOL inference engine.

Figure 8 shows the comparison results of the precision rates of the four systems. We varied the number of selected friends K for each node from 5 to 20 with 5 increase in each step. We can see from the figure that SOS has the highest query precision rate. This is because SOS can accurately identify the potential answerers based on their social information and relationship to the question and the asker. Since SOS-w/o-infer has much less parsed information than SOS-w/-infer, SOS-w/-infer outperforms SOS-w/o-infer. In Flooding, a node forwards a question to all of its friends in the social network, so the precision rate is relative low. In Random walk, the queries are randomly sent to users, so it generates low precision rate. It is interesting to find that the precision is decreased in SOS-w/o-infer and SOS-w/-infer as the K increases from 5 to 20 while the precision remains constant in Flooding and Random walk. This is because some selected questions in the Yahoo!Answers data set have very few potential answers in Facebook user profiles. As we choose more neighbors to send an asker's question, more users that cannot answer the question will receive the questions. This contributes to the decrease in the precision rate of SOS. As K increases, Random walk performs similar to Flooding as more users that are unable to answer a question receive the question.

Figure 9 shows the comparison results of the recall rates of the four systems. Since more users in the social network receive questions, the number of relevant users that receive questions increases. That is why as the number of selected neighbors K increases, the recall rate of Flooding, SOS, and Random walk increases. Since Flooding forwards a question to all the neighbors of a node in the social network, Flooding has the highest recall rate. Although SOS can find relevant answerers with high precision, it may not be able to identify all the relevant users. Therefore, SOS generates much lower recall than Flooding. Again, SOS-w/-infer outperforms SOS-w/o-infer slightly because of its prediction ability. As a user randomly approaches others for answers in Random walk, its recall rate is even lower than Flooding and SOS.

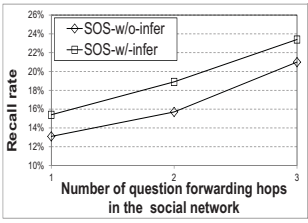


Fig. 12: Query recall with social closeness.

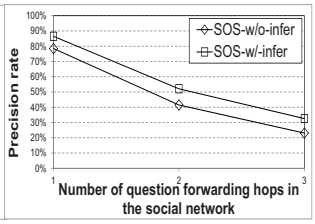
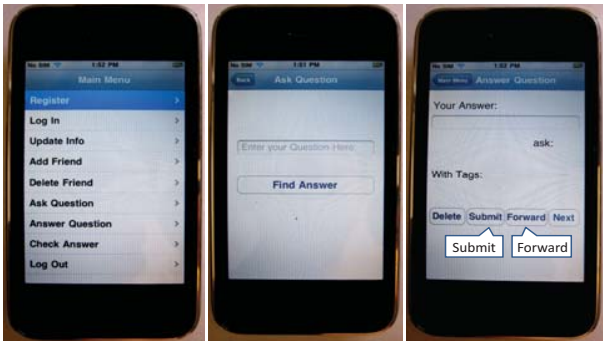


Fig. 13: Query precision with social closeness.

To evaluate the scalability of the SOS system, we test the SOS system with different network sizes. For the network with size N, we selected the first N crawled users in the data crawling step to ensure the evaluated social network of these N users has the same topology as the online social networks. Figure 10 shows how overhead, measured by the number of queries, with the number of nodes in the social network. Flooding forwards a question to all the neighbors of an asker in the social network, leading to a high overhead, although it can improve its recall rate significantly. Random walk has lower overhead than Flooding, but it still has much more overhead than SOS as it needs to keep on forwarding to randomly chosen nodes until it finds an answerer. With high precision, SOS only needs to forward questions to only a few users. Therefore, the overheads of SOS-w/o-infer and SOS-w/-infer are much less than Flooding and Random walk. We can also see from the figure that as the network size increases, the overhead in Flooding and Random walk increases, but the overhead in SOS remains constant. In SOS, most answers can be accurately located. Therefore, the node increase does not increase question forwarding hops.

Figure 11 shows the comparison results of question response delay versus network size of the four systems. We set the response delay in each hop in the social network as 12 min as shown in our trace data in Twitter. Since Flooding forwards askers' questions to all the neighbors in the social network, any neighbor with the knowledge of the question can answer the question immediately. Therefore, it generates the lowest response delay. As SOS-w/o-infer and SOS-w/-infer can accurately identify the potential answerers, the average reply delay in SOS is comparable to Flooding. Since SOS-w/-infer generates much more inferred information than SOS-w/o-infer for potential answerer selection, SOS-w/-infer produces much less response delay than SOS-w/o-infer. In Random walk, randomly selected friends have low probability to answer the question, thus a node needs to search more nodes to reach an answerer. As a result, the delay of Random walk is relatively high compared to the other two methods. We can also see from the figure that as the network size increases, the delay of Random walk increases slightly and the delay of SOS and Flooding keeps constant due to the same reasons as in Figure 10. Though Flooding provides constant response delay, this comes at a high cost of flooding overhead, while SOS generates considerably lower overhead.

To test the impact of question forwarding distance on the Q&A performance, we varied the number of social hops that a question is forwarded from 1 to 3. Figure 12 shows the recall rates of SOS-w/o-infer and SOS-w/-infer versus the different number of social hops that a question is forwarded. As the number of social hops increases, the recall rate of both SOS-w/o-infer and SOS-w/-infer increases. This is because the increased social hops lead to more users to be visited. Therefore, more relevant answers will be received. Figure 13 shows the comparison results of precision rates of the systems versus the different number of social hops that a question is forwarded. We can see from the figure that 80% of the answers can be retrieved from the direct friends. This is because in social networks, friends with a close social relationship are closely clustered. Therefore, it is very easy for a node to find the answers from the direct friends. The reason why SOS-w/-infer has both a higher recall and precision rate than SOS-w/o-infer is because SOS-w/-infer can identify the potential answerers with higher accuracy with its inferred information from users and questions.



(a) Main menu. (b) Question. (c) Answer.

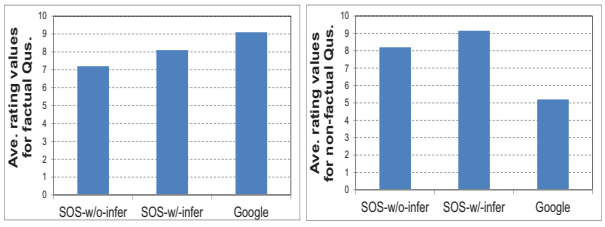
Fig. 14: Client software execution on iPhones

V. Prototype Implementation and Testing

We deployed SOS client in Object-C with iOS 4.1, and the server was written in Java using JDBC connector with MySQL. The client was deployed on iTouch/iPhones connecting to a sqlite database. The iTouch/iPhones use WiFi connectivity to access to the registration server and communicate between each other. We also developed a forum written by PHP connecting to MySQL, for the Apache2 web server, which is aimed to receive the unsolved questions from mobile users. Screenshots of the iPhone clients are presented in Figure 14. Figure 14 (a) shows the main menu of the SOS. Users can communicate with each other using the ask/answer interface and conduct operations for registration, log in or off, add/delete friends and update profile information. Figure 14 (b) and Figure 14 (c) show the question and answer interfaces, respectively. In the question interface, users type their questions in the textfield and send the questions out by pressing *Find Answer* button. In the answer interface, if a user can answer the received question, s(he) can directly submit the answers by pressing the submit button. Otherwise, s(he) can forward the questions to his/her own social friends by pressing the forward button. SOS then forwards the question to the user's K top friends.

We tested the system within a small group of 30 students in Clemson University. The students are from 6 different departments with students in natural science majors and social science majors. In the experiments, we mainly focused on four categories of questions: Music, Book, Movie and Television. We imported 9787 fuzzy keywords from WordNet that are related to the four categories into SOS's fuzzy database and imported 137 rules into SOS's rule-based inference engine for interest inference. These rules are designed based on common sense relation between personal interests. In total, 389 questions were collected from the testing. We set K to 3 considering the small size of our social network in testing.

For each question an asker asked, the question is sent to the social network via both the SOS-w/-infer system and SOS-w/o-infer system. After receiving an answer, an asker needed to rate the answer with a 0-10 star. The asker was also required to search the question's answer through Google and rate the Google results. This is to compare the performance of SOS and Google search engine. Figure 15(a) and Figure 15(b) show the comparison results of average rating values for factual and non-factual questions for SOS-w/-infer, SOS-w/o-infer and Google. The figures show that for the factual based questions such as "Who is the director of Kung Fu Panda 2?", "Who is the author of Gone with the Wind?", Google has slightly higher ratings than SOS-w/o-infer and SOS-w/-infer. This is because the number of the participants in the testing group is limited, leading to limited knowledge in our social network. The participants may not remember some of the facts asked in the factual questions. However, for some common factual questions such as "What is the oldest book in the world?", or technical questions that relate to their majors or career, such as "The author of the book Introduction to Algorithms?", the participants can give accurate answers.



(a) Factual based questions. (b) Non-factual based questions.

Fig. 15: Comparison of three systems.

On the other hand, as shown in Figure 15(b), for the non-factual questions such as "How is the course ECE613 in Clemson University?","How to set the width of caption in Latex?", SOS-w/o-infer and SOS-w/-infer have much higher ratings than Google, since the answers to these questions cannot be found in Google easily. For questions such as "Is the movie Transformer 3 worth going to watch?", which can be found in both Google and SOS, SOS even receives much higher ratings than Google, because users tend to trust the answers from their friends, and the results from Google are often overwhelmed by advertisements. From both figures, we see that SOS-w/-infer outperforms SOS-w/o-infer for both factual question and non-factual questions. This is because that the inference engine in SOS-w/-infer can provide more interest information for answerer identification, which increases the likelihood to identify a potential answerer. The results are in line with our trace-driven simulation results. From these figures, we can see that SOS provides a good user experience for information search, especially for those non-factual questions that cannot be well answered by Google. Another interesting finding from the test is that some people even directly used SOS as a communication tool as in current online social networks. For example, for some complex questions such as "How to analyze the complexity of a new algorithm? ", the answerers directly asked the asker to go to his/her office for discussion. Some users asked the question "Who want to play soccer on this coming Friday."

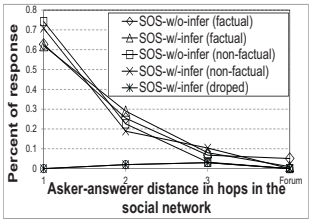


Fig. 16: Percent of responses vs. social hops.

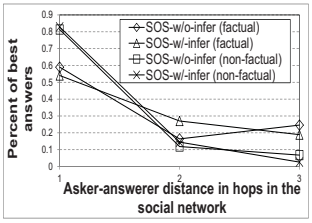


Fig. 17: Percent of best answers vs. social hops.

Figure 16 shows the percent of question responses versus the number of social hops between the asker and answerer in the social network, and the percent of dropped questions versus the number of hops between the asker and the user who dropped the question. The figure shows that the question dropping rate increases slightly at the second hop and the third hop. However, the overall packet dropping rate is still extremely small. This is because as the questions are propagated among socially close friends, users do not easily drop questions considering the close social relationship. The slightly increased dropping rate may be resulted from the decreased social closeness between the asker and question receiver. We can also see from the figure that as the number of hops in the social network increases, the number of question responses is reduced. The result indicates that socially closer friends of a user are more likely to answer the questions from the user. The figure also shows that users within one social hop are better at answering non-factual questions than factual questions because the small number of users within one-hop social distance may not have enough knowledge to answer certain kinds of factual questions. As the social hop increases, those unsolved questions can be answered by people in other social clusters that are specialized in other topics.

Each asker in the test chose the best answer for each of his/her questions. Figure 17 shows the percent of best answers to the questions given by the answerers in different social distances from the askers in the social network. The figure shows that for the non-factual questions, more than 80% of the questions can be answered by the users within one hop in the social network in both SOS-w/o-infer and SOS-w/-infer. Less than 10% of the best answers come from the users within 3 hops. The figure indicates that as the non-factual questions are normally the questions about suggestions, advises and recommendations, the answers from socially close

friends are more likely to be trusted. The figure also shows that for the factual questions, less than 60% of the best answers are provided by the friends within 1 hop. For SOS-w/o-infer, friends in three hops can provide better answers than friends in two hops. This is because factual questions need more specific knowledge to be answered. The users who have specific knowledge may be socially far away from the askers. SOS-w/o-infer can more accurately identify the best answerers with more inferred interests from the friends and the question.

VI. Conclusion

In this paper, we have described the design and implementation of a distributed Social-based mObile Q&A System (SOS). SOS is novel in that it achieves lightweight distributed answerer search, while still enabling a node to accurately identify its friends that can answer a question. SOS uses the FOL representation and inference engine to derive the interests of questions, and interests of users based on user social information. A node considers both its friend's parsed interests and social closeness in determining the friend's similarity value, which measures both the capability and willingness of the friend to answer/forward a question. Compared to the centralized social network based Q&A systems that suffer from traffic congestions and high server bandwidth cost, SOS is a fully distributed system in which each node makes local decision on question forwarding. Compared to broadcasting, SOS generates much less overhead with its limited question forwardings. Since each user belongs to several social clusters, by locally selecting most potential answerers, the question is very likely to be forwarded to an answerer that can provide an answerer. The low computation cost makes the system suitable for low-end mobile devices. We conducted extensive trace-driven simulations and implemented the system on iPhone/iTouch mobile devices. The results show that SOS can accurately identifies answerers that are able to answer questions. Also, SOS earns high user satisfaction ratings on answering both factual and non-factual questions. In the future, we will release the application in the App Store and study the Q&A behaviors of users in a larger-scale social network.

Acknowledgements

This research was supported in part by U.S. NSF grants OCI-1064230, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, CSR-2008826, CSR-2008827, Microsoft Research Faculty Fellowship 8300751, and Oak Ridge Award 2008833.

References

[1] Google. http://www.google.com.
[2] Bing. http://www.bing.com.
[3] B. M. Evans and E. H. Chi. An elaborated model of social search. *Information Processing & Management*, 2009.
[4] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. Phoaks: A system for sharing recommendations. *Comm. of the ACM*, 1997.
[5] M. S. Ackerman. Augmenting organizational memory: a field study of answer garden. *ACM TOIS*, 1998.
[6] L. G. Terveen, P. G. Selfridge, and M. D. Long. Living design memory: Framework, implementation, lessons learned. *Human-Computer Interaction*, 1995.
[7] E. Amitay, D. Carmel, N. Har'El, S. Ofek-Koifman, A. Soffer, S. Yogev, and N. Golbandi. Social search and discovery using a unified approach. In *Proc. of HT*, 2009.
[8] D. Carmel, N. Zwerdling, I. Guy, S. Ofek-Koifman, N. Har'el, I. Ronen, E. Uziel, S. Yogev, and S. Chernov. Personalized social search based on the user's social network. In *Proc. of CIKM*, 2009.
[9] S. Kolay and A. Dasdan. The value of socially tagged URLs for a search engine. In *Proc. of WWW*, 2009.
[10] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *Proc. of WWW*, 2007.
[11] H. H. Chen, L. Gou, X. Zhang, and C. L. Giles. Collabseer: A search engine for collaboration discovery. In *Proc. of JCDL*, 2011.
[12] C. Y. Lin, N. Cao, S. X. Liu, S. Papadimitriou, J. Sun, and X. Yan. Smallblue: Social network analysis for expertise search and collective intelligence. In *Proc. of ICDE*, 2009.
[13] H. Kautz, B. Selman, and M. Shah. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 1997.
[14] D. W. McDonald and M. S. Ackerman. Expertise recommender: a flexible recommendation system and architecture. In *Proc. of CSCW*, 2000.
[15] Yahoo answer. http://answers.yahoo.com.
[16] Ask.com. http://www.ask.com.
[17] F. Harper, D. Raban, S. Rafaeli, and J. Konstan. Predictors of answer quality in online Q&A sites. In *Proc. of SIGCHI*, 2008.
[18] M. R. Morris, J. Teevan, and K. Panovich. What do people ask their social networks, and why?: a survey study of status message q&a behavior. In *Proc. of CHI*, 2010.
[19] J. Teevan, M. R. Morris, and K. Panovich. Factors affecting response quantity, quality, and speed for questions asked via social network status messages. In *Proc. of AAAI*, 2011.
[20] R. W. White, M. Richardson, and Y. Liu. Effects of community size and contact rate in synchronous social q&a. 2011.
[21] M. Richardson and R. W. White. Supporting synchronous social q&a throughout the question lifecycle. In *Proc. of WWW*, 2011.
[22] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *Proc. of WWW*, 2010.
[23] Twitter. http://www.twitter.com/.
[24] Mobile internet stats roundup. http://econsultancy.com/us/blog.
[25] Facebook may be growing too fast. http://techcrunch.com/.
[26] A. Mtibaa, M. May, C. Diot, and M. Ammar. Peoplerank: Social opportunistic forwarding. In *Proc. of infocom*, 2010.
[27] R. M. Smullyan. *First-order logic*. Dover Publications, 1995.
[28] K. Toutanova and C. D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proc. of SIGDAT*, 2000.
[29] C. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. The MIT Press, June 18, 1999.
[30] M. Kirsten and S. Wrobel. Extending K-Means Clustering to First-Order Representations. In *Proc. of ICILP*, 2000.
[31] Z. Li, H. Shen, and K. Sapra. Leveraging Social Networks to Combat Collusion in Reputation Systems for Peer-to-Peer Networks. In *Proc. of IPDPS*, 2011.
[32] Z. Li and H. Shen. SOAP: A social network aided personalized and effective spam filter to clean your E-mail box. In *Proc. of INFOCOM*, 2011.
[33] C. Binzel and D. Fehr. How Social Distance Affects Trust and Cooperation: Experimental Evidence from A Slum. In *Proc. of ERF*, 2009.
[34] G. Swamynathan, C. Wilson, B. Boe, K. C. Almeroth, and B. Y. Zhao. Can Social Networks Improve e-Commerce: a Study on Social Marketplaces. In *Proc. of WOSN*, 2008.
[35] Readwriteweb. http://twitter.com/#RWW.