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

Ze Li, *Student Member, IEEE* ^{1}, Haiying Shen, *Member, IEEE* ^{1*}, Guoxin Liu, *Student Member, IEEE* ^{1} Jin Li, *Senior Member, IEEE* ^{2}

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.

Index Terms—Question and answer systems; Online social networks; Peer to peer networks

1 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 question and answer 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 question and answer (Q&A) sites such as Yahoo!Answers [15] and Ask.com [16] provide high-quality answers [17] and have been increasingly popular.

To enhance the asker satisfaction on the Q&A sites, recently, emerging research efforts have been focused on social network based 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. 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. In the centralized systems [20–22], since the centralized server constructs and maintains the social network of each user, it searches the potential 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 computing overhead to the friends. Also, it 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. Further, broadcasting to a large number of friends cannot guarantee the quality

- * *Corresponding Author. Email: shenh@clemson.edu; Phone: (864) 656 5931; Fax: (864) 656 5910.*
- ^{1} *The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634. E-mail: {zel, shenh, guoxinl}@clemson.edu*
- ^{2} *The author is with Microsoft Research, Redmond, WA 98052. E-mail: jinl@microsoft.com*

of the answers. The centralized methods, 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 answer the questions if they can or forward the question to their friends. 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 the 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 the 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 and trustable [27].

In a nutshell, 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 of this work 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 use *answer quality* to represent both the willingness of a node to answer another node's questions and the quality of its answers. We propose a method that considers both interest similarity and answer quality based on past experience in question forwarder selection in order to increase the likelihood of the receiver to answer/forward the question.
(4) We have studied our crawled data from Yahoo!Answer and Twitter with regards to node interactions in online Q&A systems and online social networks. We then have conducted extensive trace-driven simulations based on the crawled data. 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, SOS gains much higher satisfaction degree for non-factual questions than Google. Also, socially close users tend to respond questions quickly.

Note that SOS still has a centralized server to support Q&A activities for questions that are difficult to find answerers in the user social network. SOS also can collect previous questions and answers in the centralized server to improve the Q&A system performance. We do not endorse a complete removal of the centralized server from the system as it still plays an important role in the system. The rest of the paper is organized as follows. Section 2 presents related work. Section 3 and Section 4 present the trace data and the design of SOS. Section 5 and Section 6 show the trace-driven simulation results and real testbed results. We conclude this paper with remarks on future work in Section 7.

2 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, online Q&A systems and peer-to-peer data search.

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 and a SocialPageRank algorithm to capture the popularity of web pages. However, the social search aims to improve the web search engine, which performs poorly in non-factual questions [18]. Chakrabarti [28] presents HubRank, which computes and indexes certain random walk fingerprints for a small fraction of nodes in the web page graph and adaptively loads these active nodes in order to further reduce the computation cost of PageRank-like algorithm.

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 experts. 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: Many 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. Bulut *et al.* [29] studied the effectiveness of employing location-based services (e.g. Foursquare) for finding appropriate people to answer a given location-based query. Evans *et al.* [30] identified three social tactics for information gathering (i.e., directed asking, public asking, and searching) and found that using these tactics in combination may lead to a more productive social search. Brent *et al.* [31] built a prototype to be embedded in a user's Facebook network that provides algorithmic answers to questions posed via Facebook status messages. Morris [27] explored the pros and cons of using a social network tool to fill an information need compared with a search engine. They found that asking from friends provides several benefits, including the delivery of personalized answers and increased confidence in the validity of the search results. Lampe, Morris and Teevan [18, 19, 32] studied how people use status messages 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.

SOS is different from Aardvark in the following aspects: first, Aardvark is still based on a centralized server for the social network information analysis and answerer identification, which may lead to high query congestion, high server bandwidth and maintenance costs. SOS is a fully distributed system in which nodes find potential answerers in a distributed manner. Second, SOS is designed for mobile devices. Therefore, considering the limited capability of mobile devices, SOS uses lightweight knowledge engineering techniques, which has much less resource requirement compared to the machine learning techniques used in Aardvark.

Peer-to-peer data search: Data search has been extensively studied in peer-to-peer networks. To improve data search efficiency, some works use a supernode structure [33–43] in which nodes send their queries to supdernodes that conduct data search for the data requesters. Some works cluster nodes based on node interests or file semantics [44–57] so that nodes can find data from their common-interest peers. Some works consider proximity [40–42, 58–69] in data search so that nodes can fetch data from their physically-close peers. The distributed search design in SOS is based on the peer-to-peer distributed data search method that is well-known by its high scalability.

3 Data Study for Potential Advantages of Social-based Q&A

In order to study 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 7559 tweets from Twitter. Since Twitter is not designed to ask questions, we chose user ReadWriteWeb [70] that has a large number of followers in order to find more tweets and user tweeting/reply interactions. Specifically, we crawled 2559 tweets posted by user ReadWriteWeb and all his/her followers and followers' followers that do not have @username and 5000 tweets that have @username between Oct. 5, 2010 and Oct. 26, 2010. In Twitter, a user A can specifically notify another user B about a tweet by adding B's @username in the tweet. User B will then receive a notification message from the system about the tweet. In the figures below, we use Twitter-AT to denote

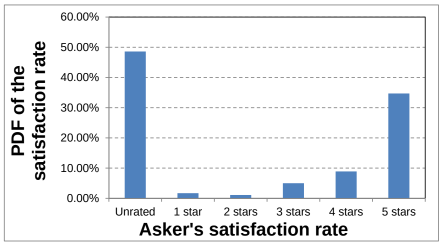


Fig. 1: Askers' satisfaction on the answers.

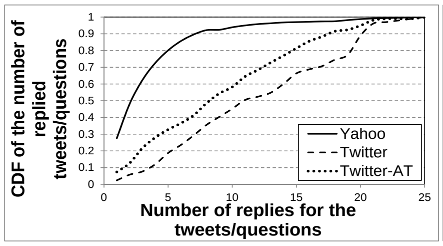


Fig. 2: Number of replies for questions.

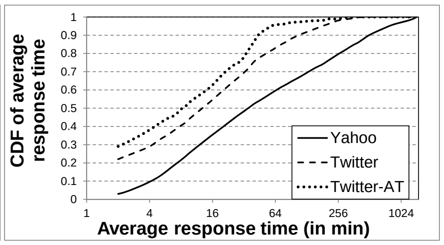


Fig. 3: CDF of average response time

the tweets with @username and use Twitter to denote tweets without @username.

First, we analyze the satisfaction of Yahoo!Answers users based on their feedbacks. In Yahoo!Answers, an asker is allowed to rate an answer with rating stars 1-5. Figure 1 shows the histogram of all the askers' satisfaction with all answers in the data set. We can see that 32% of the answerers receive 5 stars and 8% of the answerers receive 4 stars. These answers take up 80% of the answers that receive ratings. The result conforms to the observations in the previous research [17] that the answers provided in Yahoo!Answers are quite satisfying. However, nearly 50% of answers did not receive ratings. We suspect that one reason is because the users in Yahoo!Answers are not closely tied, and they may visit Yahoo!Answers only when they need to ask questions. Also, users may not take the questions or answers very seriously. Some askers may even forget to check the answers. If a question is not answered when it is in the first few pages, it may never be answered later.

Figure 2 further shows the Cumulative distribution function (CDF) of the number of replies for the replied questions in Yahoo!Answers and all crawled tweets in Twitter. In Yahoo!Answers, nearly 73% of all the questions receive more than one response, in contrast to Twitter, where more than 95% tweets receive more than one response. Also, less than 20% of all the questions receive more than 5 answers in Yahoo!Answers, in contrast to Twitter, where more than 80% of the tweets receive more than 5 responses. This is because the users in Yahoo!Answer do not have social relationship, thus they may not take others' questions as seriously as in social networks. Also, too many questions are posted on the forum every day, so it is not easy for a person who may be able to answer the question to find the questions in the first place. In Twitter, the tweets of users are only pushed to their friends that are connected by their interests and social relationships. Therefore, it is more likely for them to interact with each other. Common interest, close social relationship and frequent contact motivate a user's friends to answer his/her questions. We can also see from the figure that about 30% of the tweets with @ receive responses less than 5, while 20% of the tweets without @ receive responses less than 5. This is because the tweets that target to a specific user will not attract discussions from other users, leading to a decreased number of responses. However, as the figure shows the users with @ in Twitter are more likely to respond to each other than the Yahoo!Answer. In Yahoo!Answer, nearly 73% of all the questions receive more than one response while in Twitter with @, more than 90% of the users receive more than one response. This is because if a node A sends a tweet specifically to another node B, node B is very likely to reply the tweet to node A. In Yahoo!Answer, users do not know each other and may not have the same incentives to reply questions as in Twitter.

Figure 3 shows the CDF of the average response time for the questions that are rated in Yahoo!Answers and tweets in Twitter. We can see that less than 30% of the questions rated in Yahoo!Answers receive answers in less than 15 minutes. In contrast, in Twitter, more than 50% of the questions can be responded within 15 minutes. This result shows the advantages of shorter response time in social-based Q&A compared to Yahoo!Answers. In social networks, as users are connected by their interests and social relationships, they are more willing to interact with each other, resulting in a low response delay. We can also see from the figure that in Twitter-AT, more than 60% of the questions can be responded within 15 minutes, the percentage of which is higher than tweets without @. This is because when a user is mentioned in a tweet, s(he) is more likely to respond as s(he) knows that the sender is expecting the reply from him/her. Considering the social tie between them, the receiver is very likely to respond the tweet in a short time.

Summary: The figures show that questions posted in online Q&A sites are likely to receive few responses with long delay, though they are a good channel to inquire information. Similar result is also found in [17], which shows that the latency for receiving a satisfying answer in an online Q&A site is high with the average equals 2:52:30 (hh:mm:ss) even when the number of the registered users is very large (290,000). This is because anonymous users in a Q&A site do not have social relationship between each other, so they may not have incentives to answer others' questions. By leveraging the close social relationship and interest similarity properties among friends in social networks, social-based Q&A systems can help to overcome the inherent problems in online Q&A sites with high response rate and low response delay, since people with similar interests or close social relationship are likely to interact with each other, especially when a user specifically sends a tweet to another user.

4 System Design

4.1 Question Routing

SOS incorporates an online social network, where nodes connect each other by their social links. As shown in Figure 4, a registration server is responsible for node registration. Each user has an interest ID, which represents his/her interest. Users sharing more common interests with an asker's question are more likely to be able to answer the question. Also, users who have

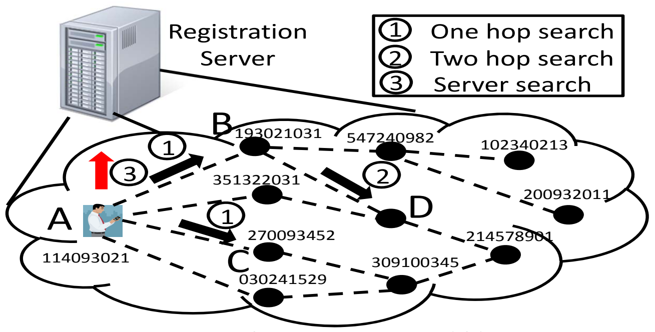


Fig. 4: Querying process in SOS.

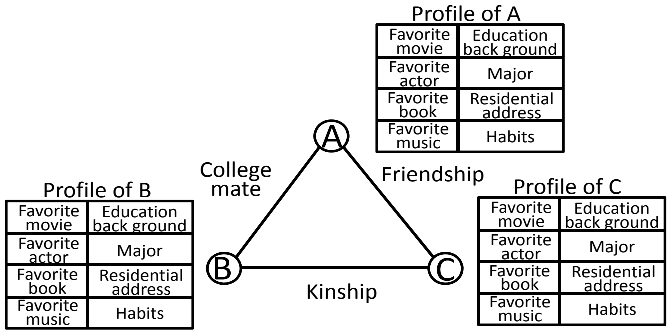


Fig. 5: An example of a node's social network.

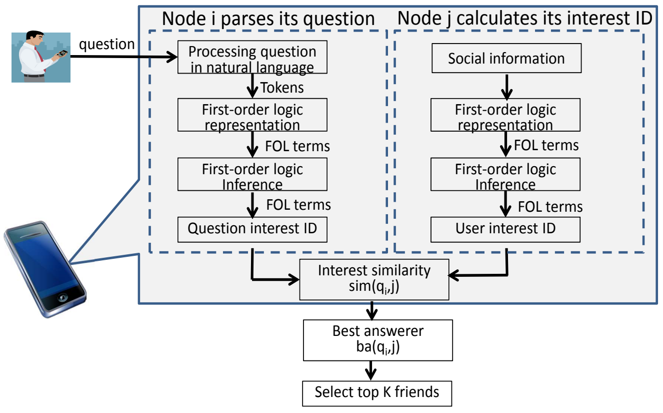

question
Node i parses its question
Node j calculates its interest ID
Processing question in natural language
Tokens
First-order logic representation
FOL terms
First-order logic Inference
FOL terms
Question interest ID
Social information
First-order logic representation
FOL terms
First-order logic Inference
FOL terms
User interest ID
Interest similarity sim(qi,j)
Best answerer ba(qi,j)
Select top K friends

Fig. 6: Answerer selection process for forwarding a question in one node.

been willing to answer questions and provided high-quality answers (measured by *answer quality*) to node i's questions previously are more likely to be willing to answer node i's questions and provide high-quality answers. Thus, SOS has a metric *best answerer* ($BA_{(q_i,j)}$) that measures the likelihood of node j to be able and willing to answer node i's question q_i with a high-quality answer. It is determined by the *interest similarity* ($S_{(q_i,j)}$) between the question q_i's interest and node j's interest as well as the *answer quality* ($Q_{(i,j)}$) of node j to node i's previous questions.

SOS defines a constant K, which is the largest number of friends that a node can send/forward a question to 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 4 shows the question routing process in SOS. In the figure, each user is associated with an interest ID. After asker A initiates a question, it forwards the question to the top K friends (nodes B and C) who have the highest *best answerer ba* values in its friend list with the question. A question receiver replies to A if it has an answer for the question. Otherwise, it 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 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 4.2)?
- How to infer the interests from a question and a user for more accurate answerer identification (Section 4.3)? 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 answer quality (Section 4.4)?

Below, we introduce the solutions for these three problems.

4.2 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 social information, the registration server recommends friends to the user, and the user then adds friends into his/her friend list. Figure 5 shows a simple example of social network, where users A, B and C are connected with each other by their social relationships. Each user locally stores her/his own profile and interest ID, and her/his friend list and their interest IDs and *answer quality* values. Each user calculates his/her own interest ID based on his/her social information and sends it to his/her friends. To calculate interest ID, as shown on the right part of Figure 6, a node first derives the first-order logic representation (FOL) [71] from its social information, then conducts first-order logic inference to infer its interests, from which it decides interest ID.

The left part of Figure 6 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 question in the nature language, and 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 question ID in the form of a numerical string. After node i parses its initiated question q_i to a question ID, it calculates interest similarity $sim(q_i, j)$ for each of its friends $j \in \mathcal{F}_i$, where $\mathcal{F}_i$ denotes the set of node i's friends. It then calculates the *best answerer* value ($ba(q_i, j)$) for each friend j by combining $sim(q_i, j)$ and *answer quality* from friend j ($qua(i, j)$). We will explain the details of calculation in Section 4.4. Finally, node i chooses top K friends that have the highest $ba(q_i, j)$ values to send the question.

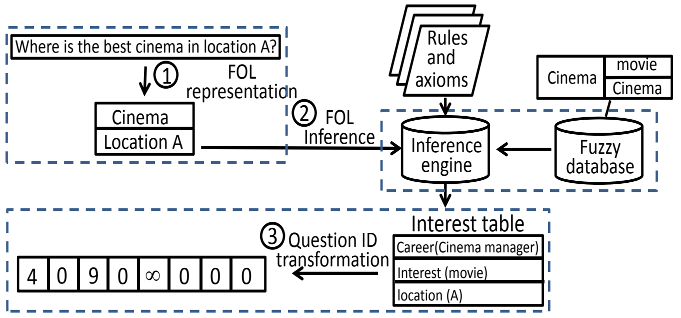


Fig. 7: An example of FOL inference for a question.

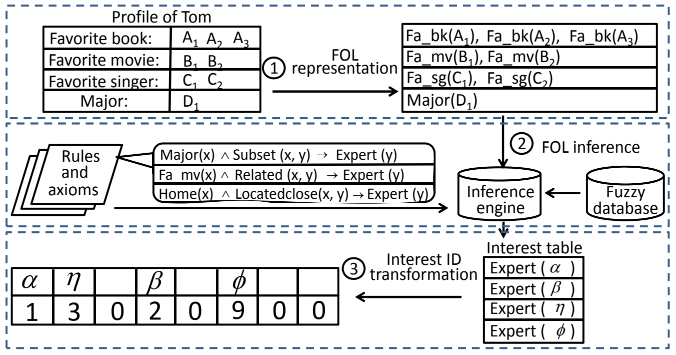

Profile of Tom
Favorite book: A_1 A_2 A_3
Favorite movie: B_1 B_2
Favorite singer: C_1 C_2
Major: D_1
① FOL representation
Fa_bk(A_1), Fa_bk(A_2), Fa_bk(A_3)
Fa_mv(B_1), Fa_mv(B_2)
Fa_sg(C_1), Fa_sg(C_2)
Major(D_1)
② FOL inference
Rules and axioms
Major(x) ∧ Subset (x, y) → Expert (y)
Fa_mv(x) ∧ Related (x, y) → Expert (y)
Home(x) ∧ Locatedclose(x, y) → Expert (y)
Inference engine
Fuzzy database
Interest table
Expert (α)
Expert (β)
Expert (η)
Expert (ϕ)
③ Interest ID transformation
α η β ϕ
1 3 0 2 0 9 0 0

Fig. 8: An example of FOL inference for a user's social information.

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 subsequently 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 interest ID 3202001001. By comparing the similarity between a question's ID and its friend's interest ID, a node can identify its friends that are able to answer questions. More details of the parsing process for a question or for a user is demonstrated in Figure 7 and Figure 8, 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.

4.2.1 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 ∧ is *connectives symbol*. In a FOL representation, connectives symbols (e.g., ∼, ∧) 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.

4.2.2 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 [72] or modern natural language processing (NLP) techniques [73] 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 are the constant symbols in the FOL representations. The step 1 in Figure 7 shows an example of FOL representation of the query. The keywords of the question "Where is the best cinema in location A?" are "cinema" and "location A".

Each node also transforms its social information into the FOL representation. Specifically, a node first represents its profile in the form of *name:value 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 can have several values. The syntax *name:value* is then transformed to the FOL representation expressed by predicate symbols *name(value)*. For example, the FOL representation of the previous example is "movie(Avatar)","movie(The Social Network)", "music(Hey, Jude)". The first step in Figure 8 shows an example of FOL representation of a node's profile. Tom has several profile information in the *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 $Fa\_bk(A_3)$.

4.3 First-order Logic Inference

As shown in the step 2 in Figure 7 and Figure 8, 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(x), related, with the information in profiles. For example, Related(cinema)=movie, Subset(computer science, algorithm), Alias(USA)=US.

The rule and axioms provide basic formulas for the inference. For example, given a rule "Major(x)∧Subset(x)=y⇒expert(y)", if Major(x_1) exists in a user's FOL inference and Subset(x_1)=y_1 exists in the fuzzy database, then we can infer that the user is 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.

Inference engine checks the rules and finds related but not obvious information. The inference engine sets each interest as an inference goal and builds lattice inference structure, as shown in Figure 9, to connect all the FOL symbols with the goals. Each node in the

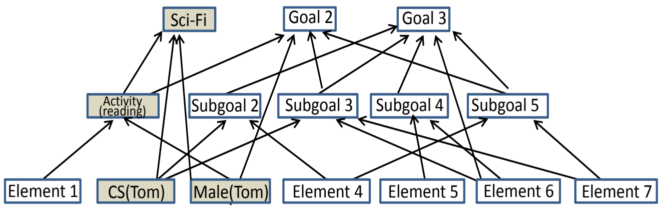


Fig. 9: Lattice in an inference engine.

lattice is a FOL syntax symbol 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 9, syntax symbols CS(Tom), Male(Tom) and Activity(Reading) all point to the goal Sci-Fi using an arrow. That is, if these three syntax symbols 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 7 and Figure 8 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 7 and Figure 8, the interests of a question (or user) are transformed to a numerical string to represent the knowledge field of a question (or a user).

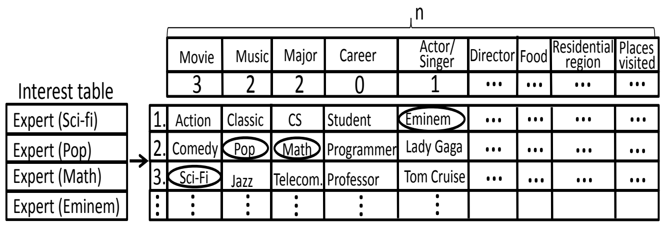


Fig. 10: An example of interest arrays.

Next, we introduce how to generate question ID and interest ID in the form of a numerical string. Figure 10 shows an example of interest arrays. The top line lists all interest categories in the system. The different interests in a category are alphabetically sorted in the category's column. Each interest in a category is represented by its entry index. For example, "Comedy" is represented by 2. Each numerical string for a question (or user) has a series of digits (e.g., 322023150). The position of each digit corresponds to each category and the value of the digit represents an interest in the category. For example, string 322023150 denotes "Sci-Fi, Pop, Math, Lady Gaga..." When generating the numerical string, each category in the table is checked. For each category, if a user/question has an interest in the category, the entry index of the interest is the digit in the corresponding position in the numeric string. Otherwise, the category's position in numeric string is set to 0. If the user/question's interest can match any specific interest in a category, the corresponding digit is set to ∞. If the user/question has several interests in a category, we use "|" to concatenate the indices of the interests in the category. For example, string 1|2|322023150 denotes "Action|Comedy|Sci-Fi, Pop, Math, Lady Gaga..." based on Figure 10. 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.

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 into the mobile phones of the users.

4.4 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 answer quality between the question sender and receiver.

4.4.1 *Interest Similarity Calculation*

To evaluate the interest similarity of a question of user i (q_i) and a user j, we use a method proposed in [74]. We use ID_{q_i} and ID_j to denote the interest strings of question q_i and user j, respectively. We use $n_{(q_i,j)}$ to denote the number of interests owned by ID_{q_i} but not by ID_j; use $l_{(q_i,j)}$ to denote the number of categories of interest elements owned both by ID_{q_i} and ID_j, and $m_{(q_i,j)}$ the number of categories of interest elements owned by ID_j but not by ID_{q_i}. Then the interest similarity of question q_i and user j is defined as:

$$\mathrm{S}_{(q_i,j)} = \frac{l_{(q_i,j)}+1}{2}\left(\frac{1}{l_{(q_i,j)}+n_{(q_i,j)}+2}+\frac{1}{l_{(q_i,j)}+m_{(q_i,j)}+2}\right) \quad (1)$$

The value of $S_{(q_i,j)}$ 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 a larger number of entries, if they share no common entries, it is more likely that they have smaller similarity than the string with a smaller number of entries and share no common entries.

4.4.2 *Answer Quality Calculation*

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 [75, 76]. However, these social closeness value calculation mechanisms are based on the whole social network topology, which are energy consuming. It is even worse when the social network dynamically changes. Therefore, the topology based social closeness calculation methods are not suitable for energy-stringent mobile devices in SOS. To reduce the load on the mobile devices, each user in SOS locally manages its first hand information on the answer quality of each of his/her friends. As the performance of the SOS largely depends on the activeness and the knowledge base of the users, user i considers the number of received answers from user j and their associated quality ratings when calculating the answer quality of user j. We call it as the feedback mechanism. Specifically, SOS initially lets users indicate the answer quality value of a newly added friend. For each received answer, an asker can rate the quality of the answer within rating scale R=[1, 5]. The answer quality value is periodically updated based on the number of answers received from friend j during each period T and the associated quality rating ($r \in [1, 5]$). For the k^{th} question sent from node i to node j, if node i receives an answer from node j during T, $x_k = 1$; otherwise, $x_k = 0$. The parameter x_k is used to represent the willingness of node j to answer questions from node i. Then, the answer quality $Q_{(i,j)}$ is calculate by:

$$Q_{(i,j)} = \alpha \cdot Q_{(i,j)} + (1-\alpha) \cdot \sum_k (x_k \cdot r_k/R) \ (x_k = 0, 1). \quad (2)$$

where $\alpha \in [0, 1]$ is a damping factor, r_k is node i's quality rating for the kth answer received from node j. A larger $Q_{(i,j)}$ implies that user j is willing and able to provide high-quality answers to user i.

Considering the high dynamism of the social networks, in which the willingness of users to answer questions and the quality of answers from a user to another user may change over time, we add a damping factor α into the answer quality calculation. In this way, the answer quality between two nodes will be sensitive to parameters r and x. That is, a node can quickly notice the active answerers who were inactive before and identify them as potential answerer to its questions.

4.4.3 Best Answerer Metric Calculation

Based on the Section 4.4.1 and Section 4.4.2, for its generated or received question q_i that it cannot answer, node i calculates the best answerer metric of each of its friends. That is,

$$BA_{(q_i,j)} = \beta S_{(q_i,j)} + (1-\beta) Q_{(i,j)}, \quad (3)$$

where $\beta \in [0, 1)$ is a parameter used to adjust the weight of the interest similarity and answer quality. Node i then selects the top K friends that have the highest $BA_{(q_i,j)}$ values and forward the question to them. We confine the TTL (number of search hops) to 3 since the social trust between two nodes decrease exponentially with distance. This relationship has been confirmed by other studies [77, 78]. Binzel *et al.* [77] discovered that a reduction in social distance between two persons significantly increases the trust between them. Swamynathan *et al.* [78] 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 process for the best answerer metric calculation and best answerer selection conducted by node i. If node i does not receive answers for its created question during the time corresponding to TTL, it resorts to the centralized server for the answers, where all user conduct Q&A activities as in online Q&A sites. Line4 - Line6 are used to periodically update answer quality of each of its friends. Line8-Line13 calculate each friend's best answerer metric and generate a list including all metric values. Line14-Line17 identify the top-K friends with the highest best answerer metric values and send the question to them. Answer quality $Q_{(i,j)}$ is pre-processed, and only interest similarity $S_{(q_i,j)}$ needs to be calculated at run time. The $sim_{(q_i,j)}$ calculation has a time complexity of $O(|\mathcal{F}_i|)$. As the number of keywords in a question is generally very small, the calculation of $sim_{(q_i,j)}$ should take a short time and costs little computation resources of the mobile devices. This top-K friend selection algorithm has a time complexity of $O(|\mathcal{F}_i|)$. Therefore, this algorithm has very low complexity considering the limited number of friends of each user.

Algorithm 1 Pseudocode of the best answerer identification executed by node i.

```
1: **Input:** $ID_i$, $ID_j$, $Q_{(i,j)}$ $(j \in \mathcal{F}_i)$
2: **Output:** top-$K$ best answerers
3: //*Periodically update* $Q_{(i,j)}$ $(j \in \mathcal{F}_i)$
4: **for** each friend $j$ in friend list $\mathcal{F}_i$ **do**
5:     Update $Q_{(i,j)}$ based on Equation (2)
6: **end for**
7: **if** create a question or receive a question it cannot answer **then**
8:     **if** TTL>0 **then**
9:         **for** each friend $j$ in friend list $\mathcal{F}_i$ **do**
10:            Calculate $S_{(q_i,j)}$ using $ID_{q_i}$ and $ID_j$ based on Equation (1)
11:            Calculate $BA_{(i,j)}$ using $Q_{(i,j)}$ and $S_{(q_i,j)}$ based on Equation (3)
12:            Add $BA_{(i,j)}$ to a list $List$
13:        **end for**
14:        QuickSort partition around the $K^{th}$ largest element in $List$
15:        Find the top-$K$ friends having the highest $BA_{(i,j)}$
16:        TTL-=1
17:        Send the question to the identified $K$ friends
18:    **end if**
19: **end if**
20: **if** does not receive answers for its created question during the time corresponding to TTL **then**
21:     Resort to the centralized server for the answers
22: **end if**
```

5 Performance Evaluation

We evaluated the SOS system using our crawled questions 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. The users are highly clustered due to the high clustering feature of the social networks.

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 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 manually selected 100 questions with keywords (e.g., movie type, directors) that can be mapped to a Facebook user's profile interests. For each of the 100 questions, we randomly assigned the answers of the question to the Facebook users whose profile interests (i.e., action movie,

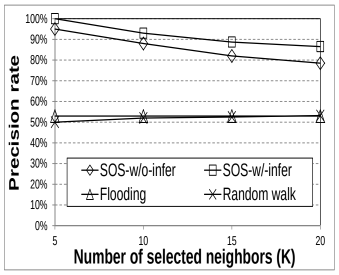


Fig. 11: Query precision rate vs. K.

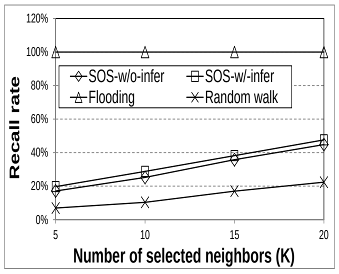


Fig. 12: Query recall rate vs. K.

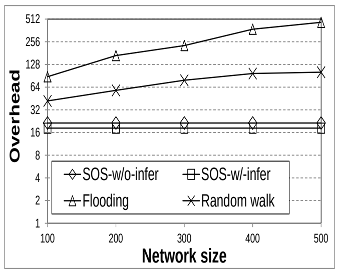


Fig. 13: Communication overhead.

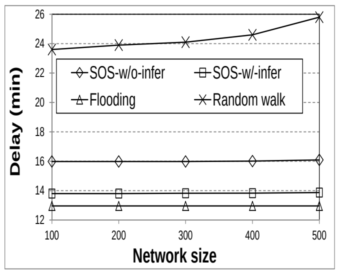


Fig. 14: Communication delay.

directors) match most to the question's keywords; one answer to one Facebook user. We use the answer quality indicated in the Yahoo!Answer as answer ratings in our experiment. The best answers have rating 5, the spam answers have rating 1 and all other answers have rating 3. The 100 questions were replicated for 10 times and randomly assigned to the Facebook users to ask. We set β to 0.5 to balance the impact of interest similarity and answer quality and set α to 0.3 by default. 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 for a certain actor A, the inference engine can find the movies and directors that are related to actor A. Then, the users who are fond of these movies or the directors are considered as the potential questions answerers. As a result, SOS with an inference engine can find more matching answerers than SOS without an inference engine.

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. 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. In the experiments, we focus on the following metrics.

- *Precision rate:* it is a measure of exactness of the returned results: $(|\text{RLA} \bigcap \text{RTA}|)/|\text{RTA}|$.
- *Recall rate:* it is a measure of completeness of the returned results: $|\text{RLA} \bigcap \text{RTA}|/|\text{RLA}|$.
- *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 in the Q&A data set from the Yahoo Q&A trace. The quality of the answers is also shown in the trace data. We use SOS-w/-infer to denote SOS with the FOL inference engine, and use SOS-w/o-infer to denote SOS without the FOL inference engine. Unless otherwise specified, we do not have feedback mechanism in SOS and the answer quality was always the initial value 1.

5.1 Comparison of the Performance of Different Systems

In this section, we compare the routing performance of SOS with the Flooding and Random walk routing methods. We use the Flooding method to represent the broadcasting-based social-based Q&A systems [17–19], in which a node forwards a question to all of its friends in the social network if it does not have the answer. In Random walk methods, a node forwards a question to K randomly selected friends if it does not have the answer. In both Flooding and Random walk methods, a node stops forwarding a question it returns an answer. The Random walk method can mimic the Q&A 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 this section, we assume all the nodes in the system are willing to answering the questions if they are able to.

Figure 11 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 12 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

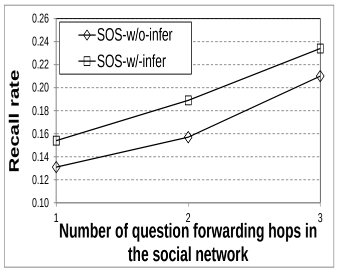


Fig. 15: Query recall with the number of social hops.

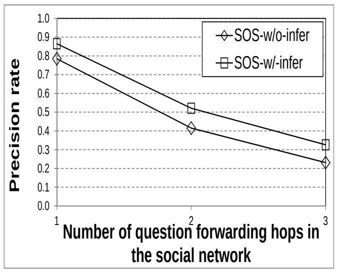


Fig. 16: Query precision with the number of social hops.

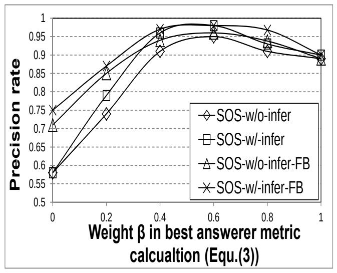


Fig. 17: Precision rate vs. scaling weight (β).

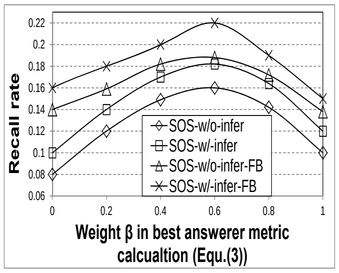


Fig. 18: Recall rate vs. scaling weight (β).

SOS-w/o-infer slightly because of its higher interest derivation ability. As a user randomly approaches others for answers in Random walk, its recall rate is even lower than Flooding and SOS.

To evaluate the scalability of the SOS system, we test its overhead measured by the number of queries with different network sizes. For the network with size N, we selected the first N crawled users in the data crawling step to ensure that the evaluated social network of these N users maintains the same topology as the real Facebook online social network. Figure 13 shows overhead 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 increase of number of nodes does not lead to the increase of the number of question forwarding hops.

Figure 14 shows the comparison results of average question response delay versus network size of the four systems. We use the medium response delay 12 min in Twitter as shown in Figure 3 as the response delay in each hop in the social network. 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 13. 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 15 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 16 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 and are likely to have common interests. 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.

5.2 Effectiveness of the Feedback Mechanism

In the experiments below, we evaluate the effectiveness of the feedback mechanism. We use SOS-w/o-infer-FB to denote SOS without the FOL inference engine but with the feedback mechanism, and use SOS-w/-infer-FB to denote SOS with FOL inference engine and feedback mechanism. The initial answer quality was set to 1. Without the feedback mechanism, $Q(i, j)$ is always 1 for each pair of nodes. In this section, the probability of a node to answer or forward a question is uniformly distributed between [p, 1]. By default, the value of p was set to 0.7. We assume that the answer quality update period is so small that it is updated once an answer is received. We assigned 1000 questions to 250 randomly selected users.

Figure 17 shows the precision rate of the received answers versus the different values of the scaling weight β. We can see that initially, the precision rate of the four systems increases as the weight β increases. When the β value passes a value around 0.5-0.6, the precision rate of the four systems decreases. A small β implies that the best answerer metric $BA_{(q_i,j)}$ largely depends on answer quality value $Q_{(i,j)}$. However, $Q_{(i,j)}$ alone cannot indicate the capability of a friend for answering the question in terms of interests. A large β implies that

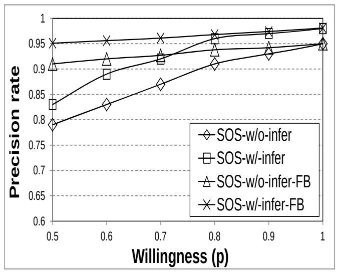


Fig. 19: Precision rate vs. willingness (p).

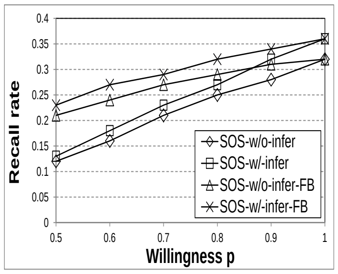


Fig. 20: Recall rate vs. willingness (p).

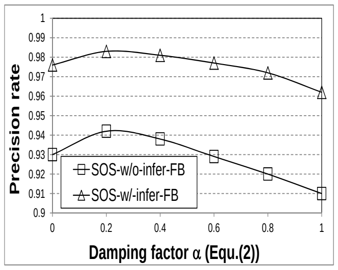


Fig. 21: Precision rate vs. damping factor (α).

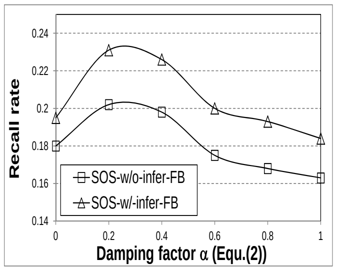


Fig. 22: Recall rate vs. damping factor (α).

$BA_{(q_i,j)}$ largely depends on the interest similarity $S_{(q_i,j)}$. However, as the quality of the answers also depends on the willingness of the users to answer/forward questions, ignoring answer quality for potential answerer selection will lead to a low recall rate. We notice that no manner SOS has the inference engine or not, SOS with FB always has higher recall rate than SOS without FB. This is because in the latter, the answer quality is solely determined by their assumed closeness with their friends, which cannot reflect the willingness of the friends to answer/forward questions. By considering the number of the received answers with their associated quality, the feedback mechanism can estimate the answer quality value that can reflect activeness and knowledge quality of their friends. We can also see from the figure that SOS with inference system has higher precision rate than SOS without inference system due to the same reason as Figure 16.

Figure 18 shows the recall rates of SOS-w/o-infer, SOS-w/-infer, SOS-w/o-infer-FB and SOS-w/-infer-FB versus different weight value β in calculating the BA metric in Equation (3). We can see from the figure that as the weight β increases, the recall rate of all of the four different systems increases first and then decreases after $\beta = 0.6$. Also, SOS with inference engine can achieve higher recall rate than SOS without inference engine. The reasons are the same as Figure 17. The figure also shows that the difference of precision rate between SOS with feedback mechanism and SOS without feedback mechanism is smaller than the difference of recall rate between these two systems.

Comparing Figure 17 and Figure 18, we see that the precision rate difference between SOS with FB and SOS without FB is small, while the recall rate difference between these two systems is large. Also, when β increases after 0.5 or 0.6, the precision rate exhibits a slight drop while the recall rate exhibits a more sharper drop. The precision rate measures how likely a returned answer is a correct answer. The recall rate measures how many correct answers in the system can be returned. Therefore, the number of dropped questions does not affect the precision rate but affect the recall rate. The feedback mechanism helps find answers with high willingness to answer questions and provide high-quality answers and avoids question droppings. Thus, it does not significantly affect precision rate but would increase recall rate. As a result, the recall rate difference between SOS with FB and SOS without FB is large while their precision rate is small. Also, as β increases after 0.5/0.6, the influence of answer quality on the answerer selection decreases based on Equation (3). Thus, the precision rate does not change much while the recall rate decreases caused by more question droppings.

Figure 19 shows the precision rates of different systems versus different willingness value p. The figure shows that the precision rate of all system increases as p increases. The reason is that a higher p value means that more users are willing to answer/forward questions from others. We can also see from the figure that even for a small willingness value p, the SOS with feedback mechanism can still achieve a high precision value. This is because the questions in SOS with feedback can be forwarded to a peer that can provide high quality answers and is willing to answer/forward answers. That is also why the increase rate of the SOS with feedback mechanism is much smaller than SOS without feedback mechanism. We can also see that SOS with the feedback mechanism has much lower increase rate than SOS without the feedback mechanism. The reason is the same as explained in Figure 18. This is because that the feedback mechanism helps avoid question droppings and find high-quality answerers even with a low p as explained in Figure 18. Thus, the precision rate of SOS with the feedback mechanism is high when p is low, and it increases slightly as p increases, while that of SOS without the feedback mechanism increases with a higher rate.

Figure 20 shows the recall rates of different systems versus different willingness value p. We can see from the figure that the recall rate of different systems increases as the willingness value p increases. The reason is that as the p increases, more nodes are willing to answer/forward questions from others. As more nodes actively participate in the Q&A activities, more high-quality answers are returned to the askers, which increases the recall rate of different systems. We can also see that no matter SOS has the inference engine or not, the recall rate of SOS with feedback mechanism is larger than the recall rate of SOS without feedback mechanism. The reason is that the feedback mechanism enables the question to be forwarded to the active nodes, avoiding the inactive users and users that provide poor-quality answers. Again, we see that SOS with the inference engine leads to higher recall rate than SOS without it no matter SOS has FB or not. This is due to the same reason as in Figure 16. Comparing Figure 20 with Figure 19, we see that recall rate increases faster than precision rate as p increases. This is due to the reason that the feedback mechanism does not significantly affect the precision rate but affects the recall rate as explained in Figure 18.

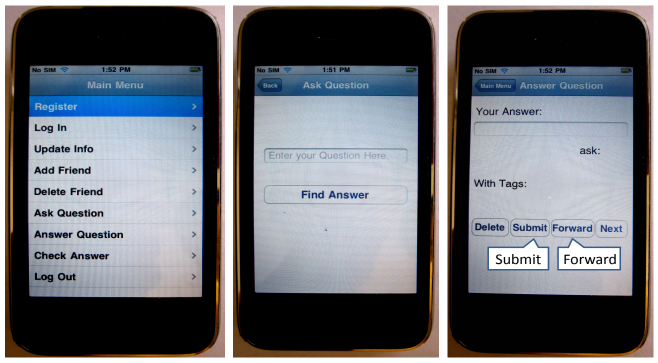
Figure 21 shows the precision rate of SOS-w/-infer-FB and SOS-w/o-infer-FB versus different values of damping factor α. The figure shows that as α increases from 0 to 0.2, the precision rates of the two systems increase initially. Once α is larger than 0.2, the precision rate decreases. When α equals 0, the value of answer quality $Q_{(i,j)}$ is dominated by recent ratings r_k, ignoring the nodes' previous behaviors. Thus, once a node returns an answer with a low quality, it has a low probability to be chosen though it returns high-quality answers previously. Therefore, the answer quality is not sufficiently accurate, leading to low precision rate. When the damping factor continues to increase after 0.2, the previous answer quality gains higher weight and the recent activeness of the users gains lower weight. Thus, the answer quality $Q_{(i,j)}$ cannot sufficiently reflect the recent activeness of the users in the social network. Therefore, $\alpha = 2$ is an optimal setting in this experiment environment that considers node past answering behavior and also assigns a relatively higher weight to node recent answering behavior.

Figure 22 shows the recall rate of SOS-w/-infer-FB and SOS-w/o-infer-FB versus different value of damping factor. We can see that the recall rate of both systems increases initially as the damping factor α increases from 0 to 0.2. After the α value is larger than 0.2, the recall rate decreases. The reason is the same as in Figure 21. From Figure 22 and Figure 21, we can see that the precision rate and recall rate of SOS-w/-infer is higher than SOS-w/o-infer. This is because the inference engine can derive more interest information as explained in Figure 11 and Figure 12.

6 Prototype Implementation and Testing

We deployed SOS client in Objective-C with iOS 4.1, and the server was written in Java using JDBC connector with MySQL. The client was deployed on iPod Touch/iPhones connecting to a sqlite database. The iPod Touch/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 23. Figure 23 (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 23 (b) and Figure 23 (c) show the question and answer interfaces, respectively. In the question interface, users type their questions in the text field 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 questions 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. The deployment lasted about 2 days and about 389 questions were asked. 0.1% of the questions were sent to the forum and 96% of the questions were answered. In the experiments, we

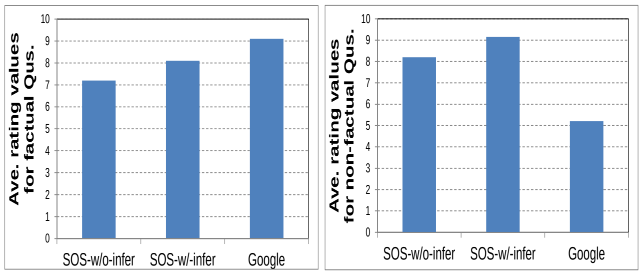


(a) Main menu. (b) Question. (c) Answer.

Fig. 23: Client software execution on iPhones

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 24(a) and Figure 24(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. 24: Comparison of three systems.

On the other hand, as shown in Figure 24(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. This result implies that users tend to trust the answers from their friends. As mentioned by a participant, "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."

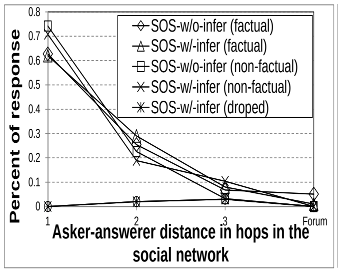


Fig. 25: Percent of responses vs. social hops.

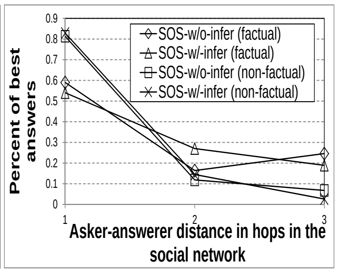


Fig. 26: Percent of best answers vs. social hops.

Figure 25 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 26 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 accepted. 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. Section 8 in supplemental material presents additional experimental results for SOS on 120 devices and computers.

7 Conclusion

In this paper, we have presented 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 answer quality 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 answerers that can provide answers. The low computation cost makes the system suitable for low-end mobile devices. We conducted extensive trace-driven simulations and implemented the system on iPod Touch/iPhone mobile devices. The results show that SOS can accurately identify 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 study the combination of SOS and cloud-based Q&A system. We will also release the application in the App Store and study the Q&A behaviors of users in a larger-scale social network.

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://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] J. Teevan M. R. Morris and K. Panovich. A comparison of information seeking using search engines and social networks. In *Proc. of ICSWM*, 2010.
[28] S. Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In *Proc. of WWW*, 2007.
[29] M. F. Bulut, Y. S. Yilmaz, and M. Demirbas. Crowdsourcing location-based queries. In *Proc. of PERCOMW*, 2011.
[30] B. M. Evans, S. Kairam, and P. Pirolli. Exploring the cognitive consequences of social search. In *Proc. of CHI EA*, 2009.
[31] M. R. Morris B. Hecht, J. Teevan and D. Liebling. Searchbuddies: Bringing search engines into the conversation. In *Proc. of CHI*, 2012.
[32] C. Lampe, J. Vitak, R. Gray, and N. B. Ellison. Perceptions of facebook's value as an information source. In *Proc. of CHI*, 2012.
[33] S. Ioannidis and P. Marbach. On the Design of Hybrid Peer-to-Peer Systems. In *Proc. of SIGMETRICS*, 2008.
[34] C. Wang and X. Li. An Effective P2P Search Scheme to Exploit File Sharing Heterogeneity. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, pages 145–157, 2007.
[35] R. Ahmed and R. Boutaba. Plexus: A Scalable Peer-to-Peer Protocol Enabling Efficient Subset Search. *IEEE/ACM Transactions on Networking (TN)*, 17:130–143, 2009.
[36] S. Seshadri and B. Cooper. Routing Queries through a Peer-to-Peer InfoBeacons Network Using Information Retrieval Techniques. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 18(12):1754–1765, 2007.
[37] M. Yang and Y. Yang. An Efficient Hybrid Peer-to-Peer System for Distributed Data Sharing. *IEEE Transactions of Computers (TC)*, 59:1158–1171, 2010.
[38] X. Shi, J. Han, Y. Liu, and L. Ni. Popularity Adaptive Search in Hybrid P2P Systems. *Journal of Parallel and Distributed Computing (JPDC)*, pages 125–134, 2009.
[39] H.-W. Ferng and I. Christanto. A globally overlaid hierarchical p2p-sip architecture with route optimization. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 22(11):1826–1833, 2011.
[40] H. Shen and K. Hwang. Locality-Preserving Clustering and Discovery of Resources in Wide-Area Distributed Computational Grids. *Transaction of Computers (TC)*, 2012.
[41] H. Shen and C. Xu. Hash-based Proximity Clustering for Efficient Load Balancing in Heterogeneous DHT Networks. *Journal of Parallel and Distributed Computing (JPDC)*, 2008.
[42] H. Shen and C. Xu. Locality-Aware and Churn-Resilient Load Balancing Algorithms in Structured Peer-to-Peer Networks. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2007.
[43] H. Chandler, H. Shen, L. Zhao, J. Stokes, and Jin Li. Toward P2P-based Multimedia Sharing in User Generated Contents. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2012.
[44] V. Carchiolo, M. Malgeri, G. Mangioni, and V. Nicosia. An Adaptive Overlay Network Inspired By Social Behavior. *Journal of Parallel and Distributed Computing (JPDC)*, 2010.
[45] A. Iamnitchi, M. Ripeanu, E. Santos-Neto, and I. Foster. The Small World of File Sharing. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 22:1120–1134, 2011.
[46] M. Li, W.-C. Lee, A. Sivasubramaniam, and J. Zhao. SSW: A Small-World-Based Overlay for Peer-to-Peer Search. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2008.
[47] E. Pagani, G. Rossi, and E. Pertoso. ORION - Ontology-based queRy routIng in Overlay Networks. *Journal of Parallel and Distributed Computing (JPDC)*, pages 28–38, 2009.
[48] G. Chen, C. P. Low, and Z. Yang. Enhancing Search Performance in Unstructured P2P Networks Based on Users' Common Interest. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, pages 821–836, 2008.
[49] G. Ruffo and R. Schifanella. A Peer-To-Peer Recommender System Based On Spontaneous Affinites. *ACM Transactions on Internet Technology (TOIT)*, 2009.
[50] K. C. J. Lin, C. P. Wang, C. F. Chou, and L. Golubchik. SocioNet: A Social-Based Multimedia Access System for Unstructured P2P Networks. *Transactions on Parallel and Distributed Systems (TPDS)*, page 1027, 2010.
[51] X. Cheng and J. Liu. NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing. In *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, 2009.
[52] Y. Li, L. Shou, and K. L. Tan. CYBER: A Community-Based Search Engine. In *Proc. of the International Conference on Peer to Peer Computing (P2P)*, 2008.
[53] A. Fast, D. Jensen, and B. N. Levine. Creating Social Networks to Improve Peer to Peer Networking. In *Proc. of KDD*, 2005.
[54] J. M. Tirado, D. Higuero, F. Isaila, J. Carretero, and A. Iamnitchi. Affinity P2P: A Self-Organizing Content-Based Locality-Aware Collaborative Peer-to-Peer Network. *Computer Networks*, 2010.
[55] R. Zhang and Y. C. Hu. Assisted Peer-to-Peer Search with Partial Indexing. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2007.
[56] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In *Proc. of the 12nd International World Wide Web Conference (WWW)*, 2003.
[57] H. Shen and C.-Z. Xu. Leveraging a Compound Graph based DHT for Multi-Attribute Range Queries with Performance Analysis. *IEEE Transactions on Computers (TC)*. 2012.
[58] Z. Li, G. Xie, and Z. Li. Efficient and Scalable Consistency Maintenance for Heterogeneous Peer-to- Peer Systems. *IEEE Transactions on Parallel and Distributed System (TPDS)*, 2008.
[59] G. A. Koenig and L. V. Kale. Optimizing Distributed Application Performance Using Dynamic Grid Topology-Aware Load Balancing. In *Proc. of IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2007.
[60] F. Lehrieder, S. Oechsner, T. Hossfeld, Z. Despotovic, W. Kellerer, and M. Michel. Can P2P-users Benefit From Locality-Awareness? In *Proc. of P2P*, 2010.
[61] F. Picconi and L. Massoulie. ISP Friend or Foe? Making P2P Live Streaming ISP-Aware. In *Proc. of ICDCS*, 2009.
[62] D. Ciullo, M.A. Garica, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek, and P. Veglia. Network awareness of P2P live streaming applications. In *Proc. of IPDPS*, 2009.
[63] D. R. Choffnes and F. E. Bustamante. Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic In P2P Systems. In *Proc. of SIGCOMM*, 2008.
[64] S. Seetharaman and M.H. Ammar. Managing Inter-Domain Traffic In The Presence of Bittorrent File-Sharing. In *Proc. of Sigmetrics*, 2008.
[65] Y. Liu, X Li, and L.M. Ni. Building a Scalable Bipartite P2P Overlay Network. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, pages 1296–1306, 2007.
[66] H. Zhang, Z. Shao, M. Chen, and K. Ramchandran. Optimal Neighbor Selection in BitTorrent-like peer-to-peer Networks. In *Proc. of SIGMETRICS*, 2011.
[67] S. Genaud and C. Rattanapoka. Large-scale Experiment Of Co-Allocation Strategies For Peer-To-Peer Supercomputing in P2P-MPI. In *Proc. of IPDPS*, 2008.
[68] H. Shen. PIRD: P2P-based Intelligent Resource Discovery in Internet-based Distributed Systems Corresponding. *Journal of Parallel and Distributed Computing (JPDC)*, 2008.
[69] H. Shen and C.-Z. Xu. Hash-based Proximity Clustering for Efficient Load Balancing in Heterogeneous DHT Networks. *Journal of Parallel and Distributed Computing (JPDC)*, 2008.
[70] Readwriteweb. http://twitter.com/#/RWW.
[71] R. M. Smullyan. *First-order logic*. Dover Publications, 1995.
[72] K. Toutanova and C. D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proc. of SIGDAT*, 2000.
[73] C. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. The MIT Press, June 18, 1999.
[74] M. Kirsten and S. Wrobel. Extending K-Means Clustering to First-Order Representations. In *Proc. of ICILP*, 2000.
[75] 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.
[76] 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.
[77] C. Binzel and D. Fehr. How Social Distance Affects Trust and Cooperation: Experimental Evidence in a Slum. In *Proc. of ERF*, 2009.
[78] E-commerce. http://en.wikipedia.org/wiki/E-commerce.

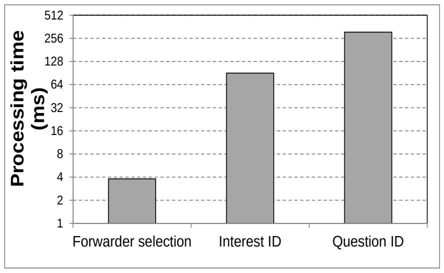


Fig. 27: Processing time.

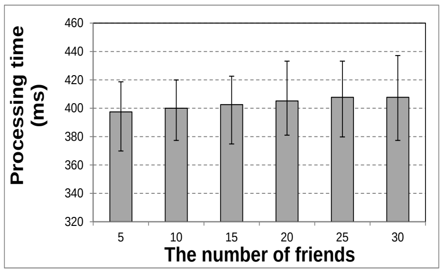


Fig. 28: Processing time vs. different number of friends.

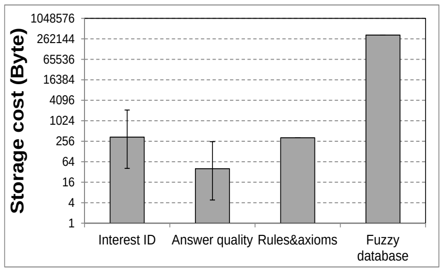


Fig. 29: Storage cost.

8 Additional Prototype Experimental Results

In this experiment, we installed SOS in 120 devices and computers. We formed the 120 nodes into a social network according to the distribution of user connections in the Facebook trace. The number of friends of the users conforms to a power law distribution. The average number of friends of a user is 10. By default, the parameter values used in this experiment are the same as in the simulation. We used 100 movie questions and their rated answers from the simulation and mapped them to the corresponding users as in the simulation. The questions assigned to each user are replicated for 10 times. We let the devices automatically initialize these questions. Each device sent out the questions following a Poisson distribution with the average rate $\lambda = 5$ by default. The experiment stopped when all questions were sent out and their TTL=0.

Figure 27 shows the average computation time for a user to generate its interest ID, to select a friend as question forwarder based on Equations (1), (2) and (3), and to generate the ID of its question. We see that the computation time for the interest ID generation is around 3.75ms, for the question ID generation is around 90.2ms, and for the forwarder selection is around 358ms, which are all very small.

We classified askers to different groups based on the number of friends of each user. We then calculated the average of the total computation time for the interest ID generation, question ID generation and forwarder selection per asker and per question in each group. Figure 28 shows the 5th percentile, 95th percentile and average total computation time of each group of users. We see that as the number of friends of a user increases, the total question computation time increases. This is because a user with more friends needs to do more calculation for forwarder selection, thus increasing latency.

Figure 29 shows the average storage cost for interest ID, answer quality of friends, rules and axioms database and inference engine per node. It also includes the 5th percentile and 95 percentile of the storage cost for interest ID and answer quality of friends. We see that the storage cost of interest ID is 339 bytes and rules and axioms database is 324 bytes. The size of answer quality is 40 bytes, and the size of the fuzzy database is 338,944 bytes. The sizes of interest ID and answer quality are only related with the number of friends of a user, and their sum equals 379 bytes for a user in a social network with average number of friends equaling to 10. In Facebook, the maximum number of friends a user is allowed to have is 5,000. Therefore, the maximum storage cost of interest ID and answer quality of a user is 1.89MB. In the test, we only focused on the inference among the movie, book, music and television categories. More rules are needed for the inference with a set of comprehensive categories (e.g., Yahoo!Answer has 26 categories). Take Yahoo!Answer as an example, since every combination of four interest categories out of total 26 categories generates rules with file size of 300 bytes and there are $A_{26}^{4} = 14,950$ combinations, so the total file size is $300B{*}14950 \approx 4$MB. The storage cost of the fuzzy word database is about 311KB. The fuzzy word database is derived from WordNet, which contains keywords from all different categories. The experimental results indicate that the storage cost of SOS will not be a burden to an average smart phone, which has storage capacity up to GBs.

Next, we compare the performance of a centralized Q&A system and a distributed Q&A system. We define the querying latency as the time period from when a question is generated to the time when a reply is received by the asker. Each device sends the questions out following a Poisson distribution with λ varying from 2.5 to 10 questions per minute with a step increase of 2.5. Figure 30 shows the 5th percentile, median and 95th percentile of querying latency versus the question generating rate λ. We created 15 threads on the server so that the server can concurrently handle 15 answers (question ID generation and forwarder selection). The figure shows that the nodes in the centralized system experience much less delay than the distributed system when the querying rate is small. However, as the querying rate increases, the querying latency in the centralized system increases while that in the decentralized system remains constant. When the querying rate equals 10, the centralized system experiences higher delay than the distributed system. Since the computation capability of devices is much lower than that of the server, to process a question, the server consumes less time. Therefore, when the querying rate is small, the querying latency in the centralized system is small. As the querying rate increases, the server receives more queries and more queries need to wait in the queue, thus increasing the average query delay. In contrast, in a distributed system, a node only receives questions from its friends. Therefore, the forwarder is unlikely to be congested. Thus, the querying latency is small even when the querying rate is large.

Figure 31 shows the 5th percentile, median and

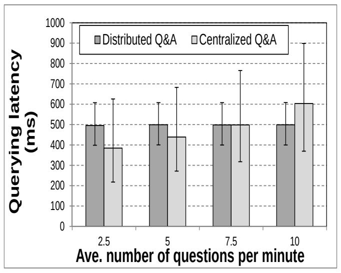


Fig. 30: Querying latency.

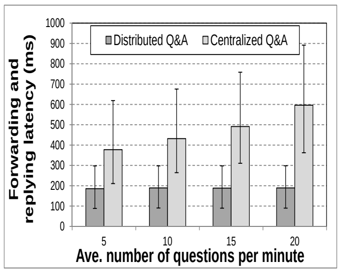


Fig. 31: Forwarding and replying latency.

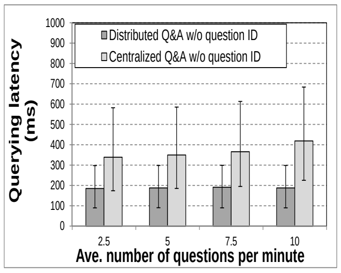


Fig. 32: Querying latency without question ID generation component.

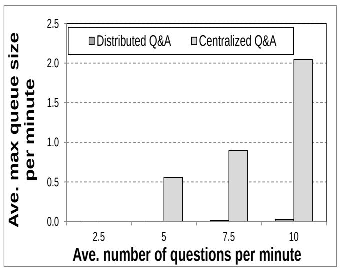


Fig. 33: Maximum queue size.

95th percentile of forwarding and replying latency (i.e., querying latency minus question ID generation time) versus the question generating rate λ. The relative performance between the centralized system and the distributed system is similar to that in Figure 30 due to the same reasons. Comparing this figure to Figure 30, we see that the centralized system decreases very slightly while the distributed system decreases dramatically. This is because the question ID generation needs much less time in the server but needs a long time in the mobile devices due to their different computing capacity.

Figure 32 further shows the querying latency without the question ID generation component. That is, each question is associated with an interest category, and is sent to friends with this category. Comparing Figure 30 and Figure 32, we find that the average querying latency in Figure 32 is smaller. This is because the systems in Figure 32 do not need to generate question IDs. We also see that the delay in the centralized system is consistently larger than the distributed system. This is because the mobile devices have less computing resources than centralized server, thus the processing latency on the mobile devices is greatly reduced by removing the question ID generation. However, the major delay in the centralized system is the queuing delay. Removing the question ID generation process does not greatly affect processing latency on the server. Since the question ID generation also generates certain latency and hence queuing latency, the centralized system produces higher latency in Figure 31 than in Figure 32.

We measured the maximum queue size of the device in the distributed system and the server in the centralized system each second, and then calculated the average of maximum queue size in each second during the whole experiment. Figure 33 shows the average maximum queue size of the devices and the server. We see that as the querying rate increases, the queue size of the server increases rapidly in the centralized system but is almost unchanged in devices in the distributed system. Since the questions in the centralized server are more likely to experience queuing delay, the question forwarding delay in the centralized system is large when querying rate of the users is high.