I.T.C

Content list available at ITC

Techno-Science Research Journal

Journal Homepage: www.itc.edu.kh

Techno-Science Research Journal

Using Cache to Optimize Question Wave Social Search Agents

Sethserey Sam[1*], Albert Tries i Mansilla[2], JosepLluís de la Rosa i Esteva[2]

[1]*Computer Science Department, Institute of Technology of Cambodia, BP 86, Bvd. Pochentong, Phnom Penh, Cambodia*
[2]*SingaporeAgents Research Lab, TECNIO Centre EASY, University of Girona, Campus de Montilivi, E17071 Girona, Catalonia (EU)*

Abstract: *This paper presents about our research in social search. Generally, the research in social search falls into two principal challenges. The first challenge is how to find more relevant answers to the question. The second one is how to increase speed in finding relevant answers. Recently, we had provided an algorithm called Question Waves (QW) to find more relevant answers compared to the baseline algorithmbreadth-first search(BFS). But, the search speed of the proposed algorithm still the subject to improve.In this paper, we introduce the agents' ability of learning the answers from the interactions with other agents so that they can quickly answer the question of other agents. We model this learning process by implementing the concept of data caching as the short-term memory of each social search agent. The result improvement of the speediness and the reduction of the number of messages used to communicate between agents, after apply agent's short-term memory concept, demonstrates the usefulness of the proposed approach.*

Keywords: Social search; query routin; BFS; Question Waves; LRU Cache

1. INTRODUCTION

Conventional methods to find relevant information, such as Web search engines, have some problems accessing content that it is not indexable, which is known as the deep Web. In 2001, the content of the deep Web was estimated to be between 400 and 550 times greater than the visible Web (Mansilla and Esteva, 2011). Although the search engines have been progressively personalized and made more contexts aware, they are less effective for typical searches (Banerjee and Basu, 2008), and their results relevance is reduced by the effects of search engine optimization (SEO). Content generated by users often belongs to the deep Web because its access is restricted by the managers of a social network and also because sometimes users only have access granted to a content portion, as is the case of online social networks, in which users can restrict the information accessibility to their acquaintances. Additionally, social network popularity is increasing; as an example, Facebook had 840 million active users at the end of 2011.

Although search engines provide fast results, they usually provide a large list of documents in which the desired information can be found, but there is not much help with searching through the list of documents provided. In that vein, Smyth et al. (2009) note that 70% of the time users are searching for information previously found by their friends or colleagues. For these reasons, there are recently developed tools, such as HeyStacks (Smyth et al., 2009) or the +1 button of Google, that contribute to the sharing of interesting results.

Considering social search, query-routing has been applied to several domains, such as Internet browsers (Wu et al., 2007), question answering (Banerjee and Basu, 2008) and recommender systems (Walter et al., 2007). The classification of the query-routing algorithms used in those domains is based on if the algorithms are unicast or multicast. In unicast query-routing algorithms, a query is

[*]Coresponding authors:
E-mail: sam.sethserey@itc.edu.kh; Tel: +855-12-509-383; Fax: +855-23-880-369

only sent to one acquaintance or user (Banerjee and Basu, 2008), while multicast algorithms are ones in which a query is sent to many of them (Wu et al., 2007; Mansilla and Esteva, 2011 and Walter et al., 2011).

Recently, we had proposed an algorithm in social search using search agents called "Question Waves" (QW) (Manilla and Esteva, 2011). QW does not only improve significantly the number of relevance answers compare to the generic Breadth-First Search (BFS) (Wu et al., 2007) but also reduces notably the number of exchange messages, which caused perturbing of searching process, between agents.

However, based on our recent experiments, QW did not give a promising speed to social search as BFS. The reason of this decrease of the speed is because QW introduce messages delay (the algorithms increase the scalability, but the time needed to satisfy a question is also increased).

To overcome this bottle-neck, we propose to provide agents the ability of learning from the answers received, and using a short-term memory model implemented as an embedded cache memory for each agent, so each agent can get the answer from their cache or from their closest friend cache quickly.

Section 2 explains our framework and a brief literature review of the two social search algorithms that we will consider in our experiments including BFS and QW. In Section 3, principles of agent's short-term memory are presented. In Section 4, simulations data is described. InSection 5, we show our experiment results. Finally, in Section 6, the conclusions and future work are presented.

2. FRAMWORK

In this paper we assume that there is an unstructured P2P social network of agents, where each agent represents its owner. The contact list of each agent consists of the agents owned by the people in the contact list of its owner. We assume that each user has only one agent, and that each agent only represents one user.

When a user ui has an information need, she requests it to her agent ai. The agent ai first will check its knowledge base, if it can satisfy the information of its owner it will do it and finish its task. Otherwise the agent will send the question to a subset of its contact list, and each of the agents that receive the message can ignore it, answer it with their knowledge base, show the question to its owner with the aim that she answers, or forward the question to a subset of its contacts (in such case the process continues, as the new receivers can perform the same tasks).

For each question, the agents can play three roles:

- **Questioner**: The questioner is the agent who started the question.
- **Answerer**: An answerer is an agent who answers a question with the agent's own knowledge or its user's knowledge.
- **Mediator**: A mediator is an agent who receives a question and forwards it to others. They also may forward an answer that they receive from another mediator or an answerer.

As data, We used the same data as described in (Manilla and Esteva, 2011) that consist on a movielens data set modified.

Collaborative Filtering Recommender Systems data, is adequate for social search simulation, as this recommender systems can be seen as Social Feedback Systems (Chi, 2009).

Collaborative filtering (CF) is used to recommend items that similar users have liked. CF is based on the social practice of exchange opinions. In the 1990s, GroupLens worked with this family of algorithms (Konstan et al., 2005; Resnick et al., 1994) leading a research line on recommender systems that expanded globally because of growing interest in the Internet. Usually, these systems rate items between 1 and 5. Pearson's correlation (Resnick et al., 1994; Shardanand and Maes, 1995) can be used to measure the similarity between users (Eq. 1).

$$u(t,v) = \frac{\sum_{i \in Im}\left(r_{t,i} - \bar{r_t}\right)\left(r_{v,i} - \bar{r_v}\right)}{\sqrt{\sum_{i \in Im}\left(r_{t,i} - \bar{r_t}\right)^2 \sum_{i \in I}\left(r_{v,i} - \bar{r_v}\right)^2}} \quad (1)$$

Where

- $u(t,v)$is the similarity of users u_t and u_v.
- $r_{t,i}$is the rating given by user $u_t$to item im_i.
- Im is the set of items.
- $\bar{r_t}$is the mean of the ratings of user u_t.

The item im_j rating $(r_{i,j})$ can be predicted with Eq. 11. In this equation, the constant K is used to match the prediction within the range of values.

$$r_{i,j} = \bar{r_i} + K \sum_{r_{k,j} \mid k \neq i} u_{i,k}\left(r_{k,j} - \bar{r_k}\right) \quad (2)$$

Milgram's experiments (Milgram, 1969), which were carried out in 1960s, consisted in a set of participants where each one was requested to deliver a document to a different target person. In case that the participant knew the target, she should send directly the document,

otherwise she must send the document to someone more likely to deliver the document to the target person. The number of participants was 296, the 73% of which followed the instructions. Each time the document was forwarded there was a probability that the receiver drops it. As result 64 folders reached their target person.

Under our point of view, Milgram's experiments can be considered as the basis of query-routing in unstructured p2p social networks. P2P query-routing algorithms can be classified as unicast and multicast. In the case of unicast query-routing, each requester sends the query to one and only one candidate. In the case of multi-cast it is sent to more candidates, and many algorithms uses as candidates all the acquaintances.

The Social Query Model (SQM) (Banerjee and Basu, 2008), allows a better understanding of unicast query-routing. The SQM is a probabilistic model that indicates the probability of obtaining a relevant answer. SQM is represented in equation 1 and considers the following parameters:

- The expertise $e_i \in [0,1]$ indicates the probability that the node a_i answers a query, with a probability 1- e_i that the node a_i forwards the query.
- The correctness $w_i \in [0,1]$ indicates the probability that the answer provided by the node a_i is correct.
- The response rate $r_{i,j} \in [0,1]$ indicates the probability that a node a_j accepts a query from node a_i. The probability that a node a_j drops the query is then 1- $r_{i,j}$.
- The policy π_j of a node a_i is a probability distribution that indicates the probability of selecting neighbours of a_i (N_i) when a_i decides to forward a query; concretely, each π_j^i indicates the probability that a_i forwards the query to a_j.

$$P_i = w_i e_i + (1 - e_i) \sum_{j \in N_i}^{n} \pi_j^i \, r_{i,j} P_j \quad (3)$$

What we can extract from SQM and Milgram's paradigm is that although it is possible to find a target person (that in the case of social search would be someone able to provide a relevant answer), the probability of finding it is really affected by $r_{i,j}$ and w_i. Furthermore if someone drops the query the answer will not be found, and the requester will not know that the search process has been stopped. In the case of multi-cast query routing, where the requester and each candidate can sent the request to many candidates, when someone drops the query the process continue.

The most usual multicast query-routing algorithm is the Breadth First Search (BFS) or flooding, it has been used for example in (Wu et al., 2007; Walter et al., 2007). In BFS the requester send the question to all her acquaintances. The receivers of the message answers or forward the question to all their acquaintances. BFS has a higher probability of answering the question, and will answer it faster, but requires a high number of messages that threaten the system scalability. With the aim to limit the query propagation, BFS often uses the Time-To-Live (TTL). TTL is initialized to certain value that indicates the maximum number of hopes, each time that a question message is forwarded the TTL value is decreased by one, and when its value is 0 it cannot be forwarded.

Recently we proposed the algorithm Question Waves (QW) (Manilla and Esteva, 2011). The idea of QW is introducing a delaying inversely proportional to the probability that the acquaintance will find the answer, the agents with a high probability of satisfying the information need will be requested as fast as possible, while the acquaintances with a low probability will be requested after a high delay. In QW each question message corresponds to an attempt of satisfying the information need. QW also introduces other mechanisms as delays in the answers with the aim to allow that further and better answers arrive first. QW not only reduces the number of messages, its main property is that the answers arrive sorted by relevance, or probability of satisfying the information need.

One of the drawbacks of QW is that the addition of delays can increment the time needed to satisfy the information need.

To overcome this bottle-neck, we propose to provide agents the ability of learning from the answers received, and using a short-term memory model implemented as an embedded cache memory for each agent, so each agent can get the answer from their cache or from their closest friend cache quickly. The detail of the proposed approach is detailed in the following section.

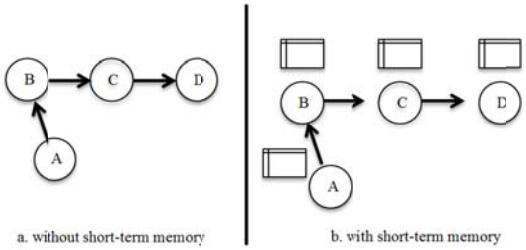

Fig. 1. Example of agent with and without short-term memory

Input	Content of LRU Cache		
A	A		
B	A	B	
C	A	B	C
D	B	C	D
C	B	D	C
A	D	C	A

Fig. 2.An example demonstrating the LRU element replacement algorithm

3. USING CACHE AS AGENT'S SHORT-TERM MEMORY

3.1. Using LRU Cache as Short-term Memory

In the previous studies (BFS (Wu et al., 2007) and Question Waves (Mansilla and Esteva, 2007); the answer stored only in the answerers so when several agents request the same question, they need to get the answer only from those answerers. This principle makes the social search slower due to query-routing duration to get the answer.

To solve this problem, we embedded a short-term memory to each agent. So when an agent receives an answer which is not in its knowledge base, it will add this answer to its short-term memory; and the agent can reuse the answer when a related question is received. The process, that each agent access to its short-term memory will be detailed in section 3.2.

Figure 1.a and 1.b illustrate the example of search agent without and with short-term memory respectively. We suppose that:

- agent D has the answer of question Q1.
- at first, agent B asked question Q1.
- later, the question Q1 was also asked by agent A.
- the access duration from one agent to another closed agent is T.

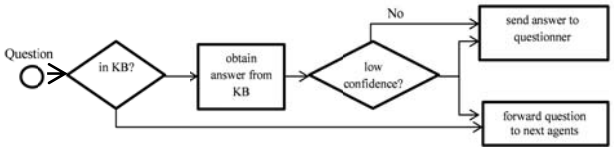

Fig. 3. The answer retrieving process of search agent without short-term memory

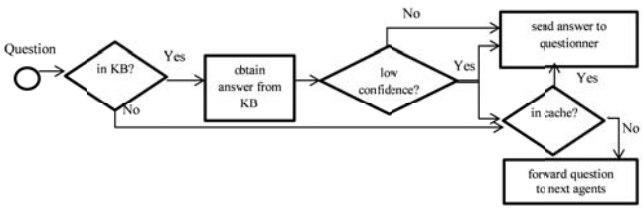

Fig. 4. The answer retrieving process of search agent with short-term memory

In the case without short-term memory (Figure 1.a), agent B and A spends 4T and 6T respectively. So the total access duration to answer the two questions from A and B is 10T. In the case with short-term memory, agent B spends 4T. During the answer transferring, the answer was stored to the short-term memory of agent C and B. So agent A spends only 2T since the answer of agent D was learned by agent B's and it is stored in its short-term memory. Thus, the agent with short-term memory help reducing 40% of access duration compared with agent without short-term memory.

The principle of the search agent's short-term memory is based on the basic and useful cache technic called least recently used cache (LRU Cache). The LRU cache evicts the element that was accessed least recently when the cache is full. Figure 2 illustrates the process of LRU Cache.

The rationale for choosing the least recently used element is, if an element has not been accessed in a while then it may not be accessed again, based in the principle of temporal locality. Each time an element is accessed we check if it is in the queue and remove it and place it at the back of the queue. Therefore, the least recently used element is always at the front of the queue. We consider that has sense to implement the short-term memory as a cache LRU, as the items accessed are refreshed by the agent, and the ones that are not in use at the end are forgotten.

In the social search case, a cache element could be composed of:

- a question
- an answer
- an answerer (an agent who answered the question)
- question effort
- date of answer

3.2. Agent's Answer Retrieving Process

When a question arrived, the ordinary search agent will look in its knowledge base (KB). If it did not find the answer, then it forwards the question to other agents. But if it find the answer, agent needs to decide whether to send its answer only (high confidence case); or to send its answer and also forward the question to other agents (low

confidence case). Figure 3 illustrates the answer retrieving process of the agent without short-term memory.

In the case that we embedded a cache to each agent, the process to retrieve the answer will be illustrated as Figure 4. The process is slidely different from the agent without cache. The answer will retrieve from the cache only when there is no answer in the agent's knowledge base or the agent did not confident in its answer.

It is important to mention that a question composed of two parameters: question value and questioner id. An answer composed of four parameters: question value, answer value, answerer id and confidential level of the answerer.

On the other hand, when an agent using cache receives an answer, it will put the answer to the cache before transferring the answer to other agents.

4. SIMULATION DATA

For our simulations we used the same data that in BFS (Wu et al., 2007) and(Mansilla and Esteva, 2007), it contains 19,463 questions distributed in 387 users.

- Questions are distributed randomly between the steps 1 and 1,000 of simulation.
- All of the simulations end at step 2,000.
- We execute 20 instances of each configuration.

5. RESULTS AND DISCUSSION

The results that we consider that should be analyzed are the number of messages, the time needed to close an answer, the precision and the recall. Figure 5 shows the time needed to close an answer. Figure 6 shows the effects of applying the short term memory in the scalability, we can see that it reduces the number of messages; also we can see that the algorithms that have a higher increase of the scalability are the algorithms that use SM. Finally Figure 7 shows the precision and the recall.

We can observe that applying a cache reduces more the time needed to close an answer and the number of messages for both BSF and QW. Meanwhile, we observed that BFS has a high drop of answer relevance using short termmemory. In the other hand, QW has a small loss of answer relevance in recall when cache is applied.

6. CONCLUSIONS

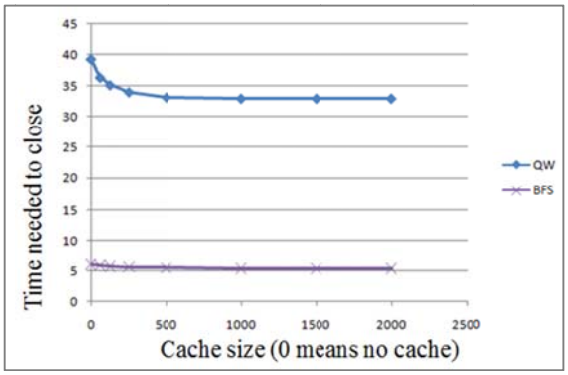

Fig.5.Time needed (in simulation steps) to close an answer in function of cache size

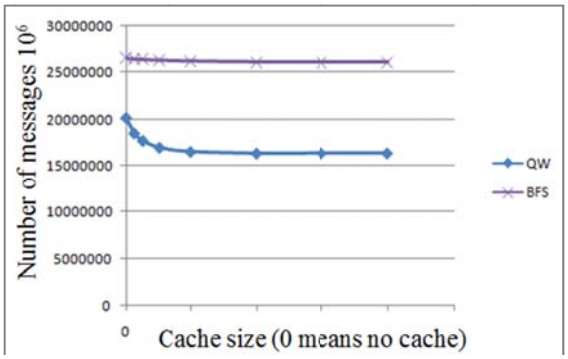

Fig.6.Number of messages generated in function of cache size

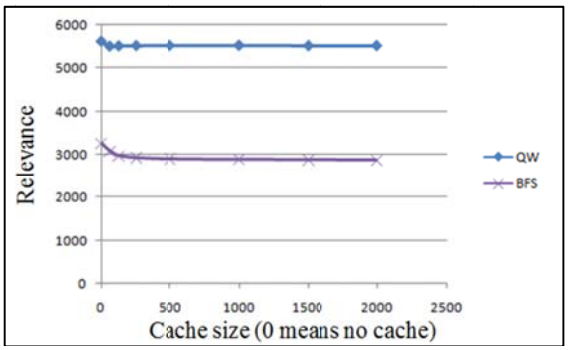

Fig.7.Answer relevance in function of cache size

From our experiments we can obtain that using short-term memory in agents reduces the time needed to satisfy aninformation need, and at the same time reduces the number of messages needed. But in the case of the algorithm BSF, there is a considerable reduction of the answer relevance.

In the case of QW the decrease of relevance is small. We consider that if the small loss of relevance is acceptable it should be better to use the cache for QW.

REFERENCES

Banerjee, A. and Basu, S.,(2008).“A social query model for decentralized search,” in … of the 2nd Workshop on Social.

Chi, E. H., “Information Seeking Can Be Social,” Computer, vol. 42, no. 3, pp. 42–46, (2009).

King, W., 1971.“Analysis of Paging Algorithms,” in Congress, pp. 485–490.

Konstan, J. A., Kapoor, N., Mcnee, S. M., and ButlerJ. T., 2005.“TechLens : Exploring the Use of Recommenders to Support Users of Digital Libraries,” no. May.

Mansilla, A. T. I and Esteva,J. L. De La Rosa I, (2011).“Asknext: An agent protocol for social search,” Information Sciences, vol. 190, no. 0, pp. 144–161,.

Mansilla, A. T. I and Esteva,J. L. De La Rosa I, (2011). “Propagation of Question Waves by Means of Trust in a Social Network,” in Flexible Query Answering Systems SE - 17, vol. 7022, pp. 186–197.

Mansilla, A. T. I and Esteva,J. L. De La Rosa I, “Question Waves: an algorithm that combines answer relevance with speediness in social search,” Information Sciences (under review).

Resnick, P., IacovouN., Suchak, M., BergstromP., and RiedlJ., (1994).“GroupLens : An Open Architecture for Collaborative Filtering of Netnews,” in Proceedings of the ACM conference on Computer supported cooperative work, vol. pp, no. 1, pp. 175–186.

Shardanand, U. and Maes, P., (1995). “Social information filtering: algorithms for automating “word of mouth,” in Proceedings of the ACM Conference on Human Factors in Computing Systems, vol. 1, pp. 210–217.

Smyth,B., Briggs,P., CoyleM., and O’MahonyM.,(2009). “Google Shared. A Case-Study in Social Search,” in User Modeling, Adaptation, and Personalization SE - 27, vol. 5535, G.-J. Houben, G. McCalla, F. Pianesi, and M. Zancanaro, Eds. Springer Berlin Heidelberg, pp. 283–294.

Walter,F. E., Battiston,S., and Schweitzer,F., (Oct. 2007).
“A model of a trust-based recommendation system on a social network,” Autonomous Agents and Multi-Agent Systems, vol. 16, no. 1, pp. 57–74.

Travers,J. and Milgram,S., (1969).“An Experimental Study of the Small World Problem,” Sociometry, vol. 32, no. 4, pp. 425–443.

Wu,L.-S., Akavipat,R., Maguitman,A. G., and Menczer,F., (2007).“Adaptive peer to peer social networks for distributed content based web search,” in Social Information Retrieval Systems: Emergent Technologies and Applications for Searching the Web Effectively, D. Goh and S. Foo, Eds. IGI Global, pp. 155–178.