BetterLife 2.0: Large-scale Social Intelligence Reasoning on Cloud

Dexter H. Hu, Yinfeng Wang, Cho-Li Wang
Department of Computer Science
The University of Hong Kong
Pokfulam Road, Hong Kong
{hyhu,yfwang,clwang}@cs.hku.hk

***Abstract*—This paper presents the design of the BetterLife 2.0 framework, which facilitates implementation of large-scale social intelligence application in cloud environment. We argued that more and more mobile social applications in pervasive computing need to be implemented this way, with a lot of user generated activities in social networking websites. We adopted the Case-based Reasoning technique to provide logical reasoning and outlined design considerations when porting a typical CBR framework jCOLIBRI2 to cloud, using Hadoop's various services (HDFS, HBase). These services allow efficient case base management (e.g. case insertion) and distribution of computational intensive jobs to speed up reasoning process more than 5 times. With the scalability merit of MapReduce, we can improve recommendation service with social network analysis that needs to handle millions of users' social activities.**

I. Introduction

The Internet has witnessed the emergence of Web 2.0 social networking sites, such as Facebook, Yelp, Amazon, and Netflix. They allow individuals to construct a public profile and articulate a list of connected users to traverse and share contents within the system. These social networking websites become very successful as they bring together social experiences from small and disconnected groups.

Further with the advancement of pervasive computing, computer access penetrates almost every aspect of our society, from GPS receiver, RFID tags to mobile devices. Many context-aware mobile applications have been developed with social networking website access. GeoLife 2.0 [1] is a recommender system of location-based social networking service. It enables users to invite people from the community to visit a place together. These places are mined from past experience of one's potential friends. PaTac [2] is platform to deliver urban, ubiquitous, personalized services for citizens and tourists. Similarly, it is also based on user's location, profile. In general, this new type of applications can enhance social interaction within ambient environment and enrich individual's choices and relationships when getting recommendation services [3]. People use these applications unconsciously to do daily tasks and leave records of experience about both people and environment. For example, a customer can comment on a meal of a restaurant on Yelp with his mobile phone, which can be viewed by other Yelp users as a reference or recommendation.

As people share more information, online communities can compile more and more digital traces. These information bear comprehensive pictures of both individual and group behaviors, which has the potential to transform the understanding of people or organizations [4]. As the degree of considering others' experience depends on the relationship, the correlation actually affects people's decision. Conventional ideas of *Social Intelligence* mainly focused on identifying the rules, norms, and modeling interactions (*e.g.*, protocols, polices) that guide appropriate behavior in a given social setting. Although this cognitive approach has served well in linguistics and artificial intelligence, its meets its limits when applied to human relationships [5].

It is usually difficult for users to judge whether certain information is useful to them or not in this huge information surge. People have to actively do search, whenever they need a piece of information. Even more effort is needed if they want to search a more trustworthy piece of information. The capacity to collect and analyze massive amounts of data generated by users demands high performance computing to accelerate knowledge discovery and to boost pattern recognition. Also a social network with n users has at most n^2 possible relations among members, the promising solution of analyzing large-scale social networks data in time is to distribute the computation workload over a large number of nodes. The cloud environment can be regarded as a massively scalable infrastructure to deliver computing (CaaS) and data (DaaS) service "anytime, anywhere" to support pervasive computing applications.

One the other hand, Case-based Reasoning (CBR) technique has long been used in context-aware recommendation systems as a reasoning technique, which is the process of solving new problems based on the solutions of similar past problems [6], [7]. Similarity-based retrieval is a beneficial feature of case-based recommenders [8]. It is applicable to problems where earlier cases are available, even when the underlying domain knowledge are not fully understood. People would benefit from a contrast-and-compare analysis by supplying a previous case and its solution to convince a user to make a decision. When the case base accumulates and the application needs to handle massive amount of user history, new reasoning platform in cloud environment need to be developed to scale with the explosion of case data.

The goal of BetterLife 2.0 is to provide an extensible framework to implement pro-active personalized recommendation

978-0-7695-4302-4/10 $26.00 © 2010 IEEE
DOI 10.1109/CloudCom.2010.108

service for users in daily life. It makes use of reasoning technique CBR and social network information to analyze large amount of data on cloud, in order to make better intelligent decisions for online and mobile users. The rest of this paper is organized as follows. We examine related work in Section II. We then detail our BetterLife 2.0 framework in Section III. Performance evaluation of a sample application is done in Section IV. Section V concludes the paper with future work directions.

II. Related Work

A. Large-scale Recommender system

Recommendation systems can be classified into four main approaches [9], namely personalized recommendation, social recommendation, item recommendation, and a combination of the three. Classical Collaborative Filtering (CF) algorithm for item recommendation has been used in most large-scale e-commerce websites (e.g. eBay, Amazon, Netflix). They try to predict utility of items based on rating of other users, especially those peers "similar" to the user. A. Das et al. proposed a scalable real time Google news recommendation engine [10] based on CF and MapReduce to guarantee the performance. Zhao et al. implemented a user-based CF algorithm on Hadoop [11] to solve the scalability problem. Zhang et al. introduced a user-centered collaborative location and activity filtering algorithm to make mobile recommendations [12] through mining knowledge from GPS trajectory. Traditional standalone CBR recommender systems will suffer similar performance degradation when large scale user cases need to be analyzed. Fortunately, MapReduce follows the divide-and-conquer strategy and facilitates the large-scale processing [13], which can be used to extend the standalone CBR application.

B. Rule-based Reasoning vs Case-base Reasoning

For traditional rule-based recommendation systems, they are not suitable for large scale social intelligence applications. Rules can be difficult to generalize or induce in certain domain without absolute standards for decision making. All triggering conditions must be strictly satisfied or exactly matched to activate the adaption, and then the system scalability is a challenge issue because the substantial rules have to be checked extensively as data accumulate. Due to the unpredictable nature of the pervasive computing environment, the intelligent agent should make decision according to the changing context in the environment. The system designer can not foresee all possible conditions or pre-determine all the rules. Failure in rule condition matching leads to malfunction of the whole system, which in turn distracts user's attention.

Also, rule-based systems are hard to maintain because rules have to be kept adding for increased size of data, while they could become more complicated. CBR is usually used when a large volume of historical data already exists, problems are not fully understood, lots of exceptions need to be considered, and service customization are needed. Therefore, in order to keep the applications in the long run, CBR is needed as it does not require additional rule generation when the data size become large. We also adopted an efficient cloud storage system like HBase to efficiently store and add new cases. The accuracy of results could be improved over time as the case base increases.

C. Social Network Analysis

Some research work has shown that at least some of the similarities within a network are caused by the influence and interactions of the people in the network. Leskovec et al. discussed the phenomenon of information cascade [14], in which individuals adopt a new action or idea due to influence by others. In some extreme cases, knowledge about a full network's behavior determines the result of its members' asking a "top hit" list available in a music downloading website.

Traditional methods for determining the importance of a node or a relationship in a network are sampling and surveying, while in a very large network the structural properties cannot be inferred by scaling up the results from small networks. To measure a user's correlation to his/her social affiliations, Berscheid et al. [15] considered Relationship Closeness Inventory (RCI) based on the degree of interdependence and determined the degree of closeness by calculating the amount of shared time, the diversity of activities, and the strength of the influence. Aron et al. [16] argued the closeness can be described as a holistic process of cognitively including another person within one's self-concept, exhibiting the Inclusion of Other in Self scale (IOS) as seven Venn-like diagrams from not overlapped at all to nearly fully overlapped.

However these two theories do not address the transitivity of correlation. Since the closeness usually reflects the mutual benefits, how to efficiently measure the social closeness to guarantee fruitful interactions is a challenging issue in large-scale social networks. The evaluation of social closeness is computational intensive, a computational social science [4] has emerged to collect and analyze data with an unprecedented breadth, depth and scale. H. Karloff et al. [17] showed the advantage of MapReduce model for a large class of PARM algorithms. Tang et al. implemented the TAP distributed leaning algorithm for analyzing social influence [18] on MapReduce to scale to real large networks model. We can also apply breadth first search or a single-source shortest paths [19] to calculate social closeness.

All in all, while other work aims at improving algorithm accuracy for recommendation in specific interest domain, BetterLife 2.0 tries to provide a sustainable, extensible and efficient framework that can generalizes the recommendation services to cover many kinds of activity interest in everyday life. Besides, while some provide recommendation services to users without justification on data trustfulness, BetterLife 2.0 considers relationship of users in social network to give more trustworthy suggestions. Lastly, some existing recommendation systems have limitations when processing large amount of data. By using Case-based Reasoning in cloud environment, this problem can be solved in BetterLife 2.0 framework.

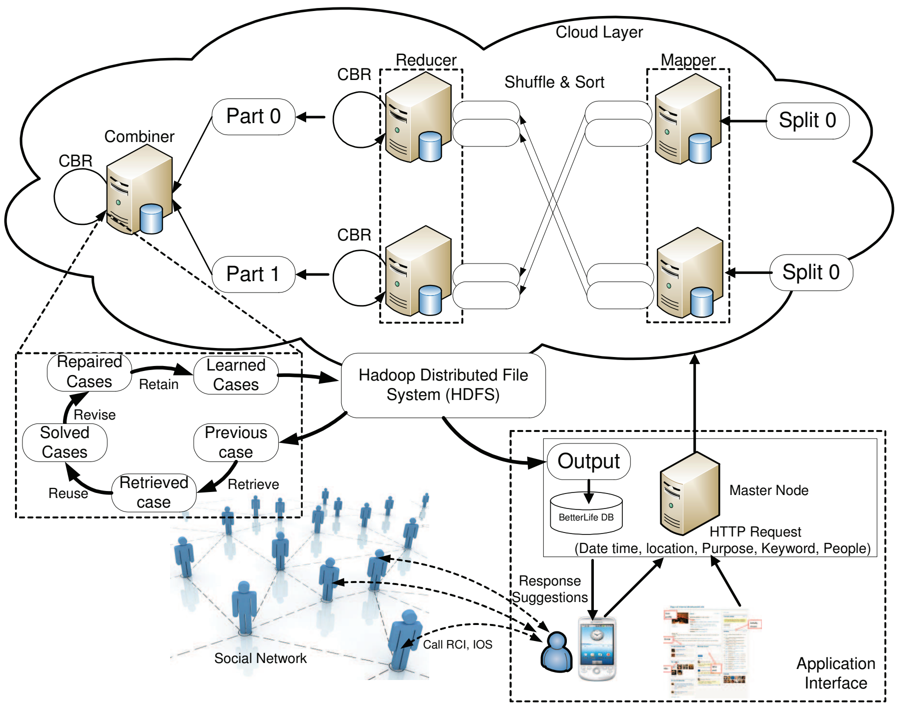

Fig. 1. Architecture of BetterLife 2.0

III. BetterLife 2.0 Overview

The architecture of BetterLife 2.0 is shown in Figure 1. There are three components within BetterLife 2.0 namely, (1) Cloud Layer (2) Case-based Reasoning Engine (3) Application Interface.

The *Cloud Layer* consists of Hadoop Distributed File System (HDFS) clusters. The Hadoop data nodes will collectively store application data represented by cases and social network information, which include relationship topology, and pairwise social closeness information. Growing user activities from online websites or mobile devices leads to fast growing data set of user history and social interactions. Traditional reasoning in web server is not adequate to handle this scale of information, as excessive I/O access and computation intensity would result in very long query response time. It is therefore necessary to have cloud facilities to handle the heavy I/O access and logic reasoning of intelligent applications. Therefore, the major bottleneck of the response time will shift from the intelligence reasoning complexity to the optimization of MapReduce functions and local computation on task nodes.

The *Case-based Reasoning Engine* extended from jCOLIBRI2 [20] has a data connector to Cloud Layer, and calculate similarity measurement between cases to retrieve the most similar ones. It can also store new case back to Cloud Layer using HBase.

The *Application Interface* uses a *master node* which is responsible for handling the request query from users. It then distributes the query to other server machines in the clusters (Map). It will also receive computation results from those server machines (Reduce). For client side, there be two types of clients. The first one is a web interface, which is extended from a social networking website for user to create profile, generate and record social interactions and edit user data, all these information will be stored into Cloud Layer. Another type of client is the mobile application on mobile phones, for user to upload or modify user context, to query and to receive server recommendations.

A. Case-Based Reasoning on MapReduce

In this section, we explains the design considerations of porting the typical CBR framework jCOLIBRI2 onto Hadoop MapReduce framework in BetterLife 2.0. As the first step, we extended jCOLIBRI2's data connector so that it can connect with the case base stored in Hadoop HDFS. In a typical CBR reasoning cycle, we have:

- *Retrieve*: Given a target problem, retrieve the most relevant or similar cases from memory to solve it. A case consists of a problem description, solution, and optionally annotations about how the solution was derived.
- *Reuse*: Map the solution from the prior case to the target problem. This may involve adapting the solution as needed to fit the new situation.
- *Revise*: Having mapped the previous solution to the target situation, test the new solution in the real world (or a simulation) and, if necessary, revise.
- *Retain*: After the solution has been successfully adapted to the target problem, store the resulting experience as a new case in memory.

There is a main difference between BetterLife 2.0 CBR engine and jCOLIBRI2 framework as shown in Figure 2. The jCOLIBRI2 CBR application would first retrieve all stored case into memory of the running machine, then operate any other operations, such as similarity matching or storing a new case. There are two problems of CBR applications developed under jCOLIBRI2 framework:

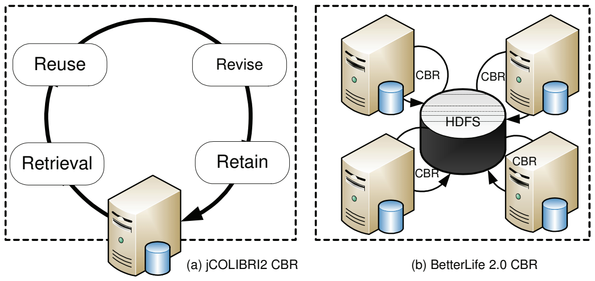

Fig. 2. jCOLIBRI2 CBR vs BetterLife 2.0 CBR

1) The computing power relied on only one single machine, which requires considerably large amount of time on computing similarity for a single query, especially for huge amount of cases stored in case base. This is in contrast with the idea of parallel computation in Hadoop MapReduce framework.
2) The main memory of a single machine is limited and therefore the heap size of JVM is also limited. Thus, for huge amount of cases stored in case base, the application cannot run as usual. So we need to run a jCOLIBRI2 CBR application in parallel, processing power and case base capacity should also be scaled up. Hence, scalability and sustainability of CBR reasoning could be achieved.

Revise and retain are basically done by mobile users. After the system recommends a solution to user, user can give feedback to system whether he/she has chosen the recommended solution. This newly formed case would be sent back to our CBR engine. We used the HBase, which is Hadoop framework's solution for storing table-like data. Since Hadoop is capable for storing huge amount of data, case indexing and case integration in the memory structure are not the concern of BetterLife 2.0's CBR engine, but rather timeliness.

To implement the CBR reasoning process on HDFS, the workflow is shown in Figure 3. The Map function can be divided into two phases, namely retrieval phase and filtering phase. In retrieval phase, data format used are designed to be as simple as possible, with a loose csv format of:
($UserID$, $Timestamp$, $Longitude$, $Latitude$, $ShopID$, $ProductID$, $Price$)

In retrieval phase, each Mapper reads their set of data locally. The reason behind is to avoid data transfer through network, which will slow down the whole MapReduce workflow. As all queries are targeted at all data (equivalently cases of CBR), no searching of data is needed in this phase. HDFS is being configured to have data replication. Failure in a data

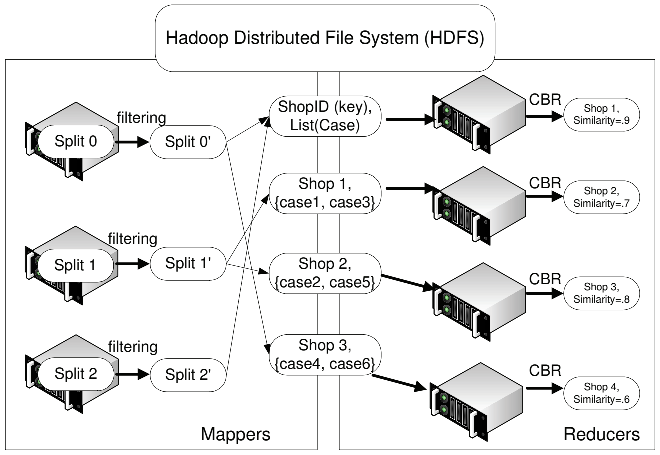

Fig. 3. MapReduce Workflow in BetterLife 2.0

node (no response from that node back to the master node) will result in calling another node with the replicated data to carry the Map function again. At the end of retrieval phase, the temporary output will be a line of string, which representing a historical case data.

The retrieved line of data is then entering the Filtering Phase, in this phase the line of data is interpreted. Information like $UserID$, $ProductID$ (barcode) are extracted from the line. Irrelevant data like mismatching $ProductID$, expired data are filtered. Unfiltered data will be calculated for similarity or further attached with the social closeness factor. These lines are then written into the intermediate output for Reducer. In our sample application, people will normally only be interested in the best result of a store, we therefore need to find the best similarity score for a particular store. We can thus use the $ShopID$ as the key and the line of data as value in the Mapper.

Each Reducer will then receive a key $ShopID$ and a collection of values (cases under that key). CBR is then performed here. For each value, that is the line of case, CBR calculates the similarity measurement of that case with the query, as in the normal retrieve phase. This phase will define similarity functions for each attribute to find k past cases which are most similar to the current query using k-nearest neighbors algorithm (K-NN). The similarity $Similarity(N,P)$ between a new case N and a past case P is calculated as follows,

$$Similarity(N,P) = \sum_{i=1}^{n} Sim(N_i,P_i) * W_i$$

$$Distance(N,P) = 1 - Similarity(N,P)$$

$$\sum_{i=1}^{n} W_i = 1$$

N_i is the value of attribute i of the new case N, P_i is the value of attribute i of the past case P, n is the number of relevant attributes in the case. $Sim(N_i,P_i)$ is the local similarity measurement between attribute i values.

The $Similarity(N, P)$ is the global similarity measurement between the two cases, which is a weighted sum of all local similarity values. Each weight W_i is in the range of [0,1]. Right now the domain knowledge will be used to set the weight of each attribute. The $Distance(N, P)$ between a new case N and a past case P is used to represent the distance between two cases considering all its attributes' similarities. In our demo application, we identified four types of similarity functions:

- **Location Similarity** The similarity of two GPS location attributes is defined as Equation (1), where *MaxDistance* is a predefined maximum distance between two points within a certain region. When $Distance(N_{gps}, P_{gps}) >$ *MaxDistance*, $Sim(N_{gps}, P_{gps}) = 0$.

$$Sim(N_{gps}, P_{gps}) = 1 - \frac{Distance(N_{gps}, P_{gps})}{MaxDistance} \quad (1)$$

- **Timestamp Similarity** This similarity is defined in Equation (2), where $Diff(N_t, P_t)$ means their relative difference in minutes within one day. Because we assume that user's behavior pattern will be similar from day to day, but could vary a lot through out a day.

$$Sim(N_t, P_t) = 1 - \frac{Diff(N_t, P_t)}{24 * 60} \quad (2)$$

- **Social Closeness Similarity** This similarity is used to incorporate interpersonal closeness into similarity measurement among cases. The closer the two users in the social closeness, the higher influence they will have on each other as defined in Equation (3):

$$\max_{p(i,j) \in P(i,j)} \prod_{e(u,v) \in p(i,j)} w(u, v) \quad (3)$$

  This equation means that we need to find out the path that maximizes the products of weights of its edges, since two online user can be connected via multiple different path, as one user may belong to multiple communities with difference social closeness. It will be explained more in Section III-B.
- **Price Similarity** This similarity returns the similarity of two prices for the same product in two cases. It uses McSherry's "Less is Better" formula

$$Sim(P_{price}, N_{price}) = \frac{Max_{price} - N_{price}}{Max_{price} - Min_{price}} \quad (4)$$

  where Max_{price}, Min_{price} are maximum and minimum prices of that product set by application, and P_{price} is not taken into account.

To make the result list from all Reducers more meaningful, at the final stage, we sorted the result list according to similarity value into one list and chop a small set which exceeds a predefined threshold $\theta = 0.8$.

B. Social Network Analysis

In order to make a more relevant recommendation system, we need to trace individual's relationship. This section explains the CBR reasoning with social network analysis. After all the cases are grouped by their user id, we will have a graph with each node corresponding to one user. We use the breadth first search (BFS) algorithm on MapReduce to calculate their closeness according to pairwise relationship weight. Figure 4 showed an example of weight propagation diagram given each edge's weight that represents the pairwise closeness of two nodes. Consider a case associated with its node, we need to find out the top k nearest cases in the whole case base, based on distance equation:

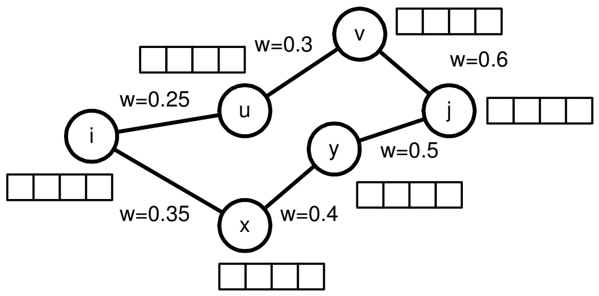

Fig. 4. Social Closeness Propagation Diagram

$$\begin{aligned} Distance(N, P) &= 1 - W_{gps} * Sim(N_{gps}, P_{gps}) \\ &- W_t * Sim(N_t, P_t) \\ &- W_{price} * Sim(N_{price}, P_{price}) \\ &- W_{social} * \max_{p(i,j) \in P(i,j)} \prod_{e(u,v) \in p(i,j)} w(u, v) \end{aligned}$$

Assuming a case C with node $n_{C.uid}$ at iteration i, its current estimated distance to given query case Q is:

$$Distance(C, Q)_i = c - W_{social} * w_1 * w_2 * \ldots * wi \quad (5)$$

where

$$\begin{aligned} c &= 1 - W_{gps} * Sim(N_{gps}, P_{gps}) \\ &\quad - W_t * Sim(N_t, P_t) \\ &\quad - W_{price} * Sim(N_{price}, P_{price}) \end{aligned}$$

and c has been calculated during previous task's Mapper phase. At iteration $i + 1$:

$$Distance(C, Q)_{i+1} = c - W_{social} * w_1 * w_2 * \ldots * w_i * w_{i+1} \quad (6)$$

So we have:

$$Distance(C, Q)_{i+1} = c - (c - Distance(C, Q)_i) * w_{i+1} \quad (7)$$

Algorithm 1 and 2 show the steps of BFS using the distance function $Distance(N, P)$ to get the most similar cases with social closeness information. Given a social network topology

Algorithm 1 SocialNetworkMapper (Key k, Node n)

```
 1: if $n$.Color == GRAY then
 2:    for all edge $e$ of $n$ do
 3:       Node $vnode \leftarrow$ new Node ($e$.ToID)
 4:       $vnode$.Distance $\leftarrow c - (c - n.Distance) * e.Weight$
 5:       $vnode$.Color $\leftarrow$ GRAY
 6:       $word \leftarrow vnode$.Id
 7:       Emit $< word, vnode >$
 8:       $n$.Color $\leftarrow$ BLACK
 9:    end for
10: end if
11: $word \leftarrow n$.Id
12: Emit $< word, n >$
```

and their pairwise closeness for each direct relationship, Algorithm 1 and 2 try to explore all the possible paths between the query user and others by node coloring (WHITE, GRAY, BLACK). All nodes will be initially be colored WHITE. Upon discovering a new node, the new node will be colored as GRAY. It will become BLACK after finishing exploring it.

Algorithm 2 SocialNetworkReducer (Key k, Iterator V)

```
 1: $distance \leftarrow$ MAX
 2: $color \leftarrow$ WHITE
 3: $edges \leftarrow$ NULL
 4: for all Node $u \in V$ do
 5:    if $u$.Edges.size $> 0$ then
 6:       $edges \leftarrow u$.Edges
 7:    end if
 8:    if $u$.Distance $< distance$ then
 9:       $distance \leftarrow u$.Distance
10:       { /* Save the minimum distance */ }
11:    end if
12:    if $u$.Color.Ordinal $> color$.Ordinal then
13:       $color \leftarrow u$.Color
14:       { /* Save the darkest color */ }
15:    end if
16: end for
17: Node $n \leftarrow$ new Node ($k$)
18: $n$.Distance $\leftarrow distance$
19: $n$.Edges $\leftarrow edges$
20: $n$.Color $\leftarrow color$
21: Emit $< k, n >$
22: if $color$ == GRAY then
23:    reporter.incrCounter(Counters.MOREGRAY, 1)
24: end if
```

IV. Performance Evaluation and Analysis

We prototyped the BetterLife 2.0 framework and developed an application of location-based price comparison to evaluate feasibility, performance and accuracy of CBR technique on cloud with MapReduce and social closeness information. This application allows mobile user to find the best place (with best price and from trusted data source) to buy things he wants from his mobile phone. Because it is always a dilemma for consumers to strike a balance between shop location and product price in that shop. The user can take a picture of the barcode to search for his intended product. The user ID, barcode information, and detected GPS location will be send to server to do analysis with trusted data according to the user's social relationship, which comes from the product rating social networking website built from Elgg [1].

TABLE I
Node Environment

CPU	2 x Intel Quad-Core E5540 Xeon CPU, 2.53GHz, 8MB cache
Memory	16GB DDR3 memory, 1066MHz, dual ranked UDIMMs
Storage	2 x 250GB 7.2K RPM SATA hard disks, running in RAID-1
Network Interface	Broadcom 5709 dual-port
OS	Fedora 11

The MapReduce cloud environment is based on Hadoop 0.20.2. We implemented an application daemon on the Hadoop's master node to accept query from client, which can be either the browser accessing our product rating website, or the mobile client implemented in HTC Magic. The Hadoop cluster environment is shown in Table I.

The experiments are designed to compare the performance of CBR engine between standalone machine and Hadoop framework, and to compare the accuracy of CBR engine with or without social network relationship analysis. We used synthetic data sets to carry out stress test and measure response time under various case base size. For performance comparison, we measured the reasoning time for processing a query, excluding the time of query/result communication between Android client and server side Application Interface.

To evaluate accuracy, we use a 10-fold cross-validation method [21] on three sets of sample data with weighted kNN algorithm. In our experiment, we set k = 1 and 3 respectively. For $k = 3$, the solution is considered as correct when it appears in the best 3 most similar cases globally. There are 103 user accounts in our product rating social networking website. We recorded some activities like commenting on product, joining groups and following friends, to demonstrate a mini-community and form historical cases. We used locations of 7-Eleven convenient stores in Hong Kong, together with the social network topology of these 103 users. To obtain enough cases under different contexts, users' behaviors were simulated by a set of pre-defined rules (*e.g.*, location clusters, product type clusters, time clusters,). The designed evaluation process is carried out on 6 data sets with different case base size. For each set of data, we randomly select one percent of cases to manually verify that solutions are reasonable in human's perspective. The Hadoop cluster consists of up to 16 nodes for slave operation and one cluster node for master operation. Slaves are responsible for data node in HDFS and will carry

[1]http://elgg.org/

out computing tasks.

A. *Query Response Time*

We first compared the query execution time on a standalone machine and on Hadoop machines (with 5 cluster nodes). The measured response time did not include network latency (to and from user's mobile device) as we only concern the speed of the reasoning engine. Table II compares query response time (in ms) of the reasoning engine on jCOLIBRI2 (standalone machine) and BetterLife 2.0 (Hadoop). For case base size = 2500K, standalone machine requires 159s to response, while Hadoop requires 29s to response, which is more than 5 times faster than standalone jCOLIBRI2. For a larger case base size, standalone machine even cannot run the reasoning engine properly. In particular, our testing machine has a main memory of 16GB, which is considerably large for a single machine. It can only hold case base size (no. of cases) as large as 15000K. This is because of the JVM heap size, due to the limited memory available. However, with Hadoop framework, the reasoning engine can run even the case base size is as large as 25000K, while the response time only scales almost linearly (to 50s).

The HDFS test has shown especially promising result. In writing a 400MB case data with 2 slave nodes enabled, HDFS takes 23 seconds to finish. The average I/O rate is about 17.4 MB/s. The read operation also shows a similar result. While in another test that writes 3GB case data with 15 slave nodes, it still takes only 23 seconds to finish. The average I/O rate is 130 MB/s. The result shows that Hadoop is performing very well in large data I/O and increase in hardware resource could gain almost linear increase in performance. However, some issue arose when testing MapReduce computing performance. Because Table II tells that Hadoop performs better than standalone machine only when the input is large enough. It should be also noted that there is a start-up delay for the MapReduce call to take place. This issue can be partially solved by using online MapReduce solution [22]. Another way of improving CBR performance is to apply domain-specific index to cases so that a request would not scan through all cases in the data set. Hadoop can be used to carry out the indexing job as an off-line operation and it is not involved in the user request workflow. A brief idea of this improvement is that Hadoop can be used in background to instantaneously build a decision tree to classify cases by criteria like price, time, location, etc. Classified cases are stored in different files and folders regarding to the built decision tree. CBR will therefore only need to find the file storing the suitable subset of cases and compute similarity among a much smaller subset.

B. *Accuracy with Social Network Analysis*

Figure 5 shows the accuracy improvement of the query result with social network analysis, using k-Nearest Neighbors algorithm with $k = 1$ and $k = 3$ respectively. When $k = 3$, accuracy in both cases is satisfactory (at least 70%). For both $k = 1$ and $k = 3$, the result accuracy is improved more than 10% with social relationships taken into consideration (labeled by 1nn-w and 3nn-w in Figure 5). This is due to the fact that when data are synthesized, we imitated some bogus users to provide product ratings in the website. As these users generally have lower social closeness with other users, their cases are less likely to be taken into consideration in case retrieval, even though their information about a product is preferable. This reflects the reality that some people intend to spam information for various reasons such as to promote his own product. For the accuracy test, it is also found that the accuracy with 1-NN retrieval method is not as high as expected.

All in all, we have demonstrated the significant improvement on both performance and accuracy of BetterLife 2.0, compared with standalone jCOLIBRI2 framework. BetterLife 2.0 framework can support a scalable reasoning engine while the response time is still in acceptable level (considering the case base size is as large as 25000K). The actual processing time of the query is within 30 seconds, which is relatively short. If the start-up cost can be further shortened by other online MapReduce solutions, the result would be more desirable.

TABLE II
TIME COMPARISON BETWEEN JCOLIBRI2 (STANDALONE) AND BETTERLIFE 2.0 (HADOOP)

No. (K)	Hadoop	Standalone	No. (K)	Hadoop	Standalone
100	23015	7121	10000	31000	700886
200	23048	13252	12500	35041	867455
500	25002	34582	15000	41023	1049880
1000	27007	69699	17500	42030	-
2500	29018	159145	20000	44030	-
5000	31078	329378	22500	47046	-
7500	33002	511856	25000	50062	-

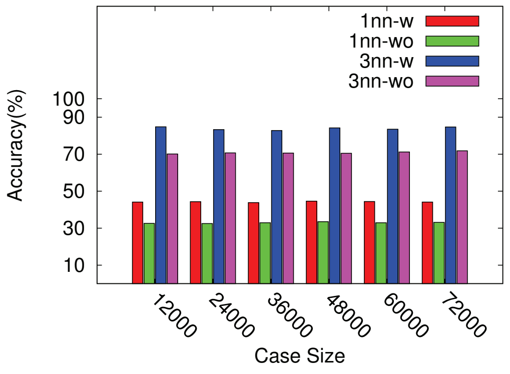

Fig. 5. Accuracy Comparison with different k and social information

V. CONCLUSION AND FUTURE WORK

We addressed various technical aspects to support large-scale intelligent recommendation service with social network analysis. The proposed framework BetterLife 2.0 is based on the Case-Based Reasoning technique for its additive knowledge space growing, and MapReduce framework for its large scale processing capability on cloud, and social network information for more relevant recommendation. Through various large-scale evaluations, we show that queries processing time using Hadoop can be highly reduced, as compared to standalone reasoning engine. Moreover, social relationship plays an important role in reasoning, selecting trustworthy data for recommendation.

Future work includes developing more applications with BetterLife 2.0 framework, covering various types of activities in daily life, such as transportation, restaurant recommendation. Also, more contexts can be collected from user's mobile devices, such as schedule, moving speed, and weather, temperature from environment. This can enhance the applicability of BetterLife 2.0 from passively answering user queries to proactively providing intelligent decision to user. We also need to further improve the timeliness of this whole process with other online MapReduce solutions.

ACKNOWLEDGMENT

This work is supported in part by Hong Kong UGC Special Equipment Grant (SEG HKU09) and China 863 grant 2006AA01A111 (CNGrid).

REFERENCES

[1] Y. Zheng, Y. Chen, X. Xie, and M. R. Asia, "GeoLife2.0: A Location-Based Social Networking Service," *2009 Tenth International Conference on Mobile Data Management Systems Services and Middleware*, no. 49, pp. 357–358, 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5088957

[2] L. Ceccaroni, V. Codina, M. Palau, and M. Pous, "Patac: Urban, ubiquitous, personalized services for citizens and tourists," in *ICDS '09: Proceedings of the 2009 Third International Conference on Digital Society*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 7–12.

[3] J. Häkkilä and J. Mäntyjärvi, "Developing design guidelines for context-aware mobile applications," in *Mobility '06: Proceedings of the 3rd international conference on Mobile technology, applications & systems*. New York, NY, USA: ACM, 2006, p. 24.

[4] D. Lazer, A. Pentland, L. Adamic, S. Aral, A.-L. Barabási, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, T. Jebara, M. Macy, D. Roy, and M. V. Alstyne, "Computational Social Science," *Science*, vol. 323, no. February, pp. 721–723, 2009. [Online]. Available: http://www.sciencemag.org/cgi/data/323/5915/721/DC1/1

[5] D. P. Goleman, *Social Intelligence: The New Science of Human Relationships*. NY: Bantam, Sep. 2006. [Online]. Available: http://www.amazon.com/Social-Intelligence-Science-Human-Relationships/dp/0553803522

[6] A. Aamodt and E. Plaza, "Case-based reasoning:foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, 1994. [Online]. Available: http://portal.acm.org/citation.cfm?id=196115

[7] F. Dong, L. Zhang, D. H. Hu, and C.-L. Wang, "A case-based component selection framework for mobile context-aware applications," *Parallel and Distributed Processing with Applications, International Symposium on*, vol. 0, pp. 366–373, 2009.

[8] D. Bridge, M. H. Göker, L. McGinty, and B. Smyth, "Case-based recommender systems," *The Knowledge Engineering Review*, vol. 20, no. 3, p. 315, 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1132830

[9] "A guide to recommender systems," http://www.readwriteweb.com/archives/recommender_systems.php, 2010.

[10] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization:scalable online collaborative filtering," *International World Wide Web Conference*, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1242572.1242610

[11] Zhi-Dan Zhao and Ming sheng Shang, "User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop," in *International Workshop on Knowledge Discovery and Data Mining*. IEEE Computer Society, 2010, pp. 478–481. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/WKDD.2010.54

[12] Vincent W. Zheng and Q. Y. Bin Cao, Yu Zheng, Xing Xie, "Collaborative Filtering Meets Mobile Recommendation: A User-Centered Approach," 2010.

[13] J. Dean and S. Ghemawat, "MapReduce:simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1327452.1327492

[14] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst, "Cascading behavior in large blog graphs: Patterns and a model," in *Society of Applied and Industrial Mathematics: Data Mining (SDM07)*, 2007.

[15] M. S. Ellen Berscheid, "Measuring closeness: The relationship closeness inventory (rci) revisited," in *The handbook of closeness and intimacy*, LAWRENCE ERLBAUM ASSOCIATES LTD. Erlbaum, 2004, pp. 81–101. [Online]. Available: http://www.questia.com/PM.qst?a=o&d=104635124

[16] A. Aron, E. N. Aron, and D. Smollan, "Inclusion of other in the self scale and the structure of interpersonal closeness," *Journal of Personality and Social Psychology*, vol. 63, no. 4, pp. 596–612, 1992.

[17] H. Karloff, "A Model of Computation for MapReduce," *Time*, pp. 938–948, 2010. [Online]. Available: http://www.siam.org/proceedings/soda/2010/SODA10_076_karloffh.pdf

[18] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," *International Conference on Knowledge Discovery and Data Mining*, pp. 807–816, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1557019.1557108

[19] H. ElGindy and C. Y. Pang, "Two Parallel Algorithms for Shortest Path Problems," in *Proc 1980 Conf on Parallel Processing*, vol. 3. Dept. Comput. Inform. Sci., Univ. Pennsylvania, 1980, pp. 244–253.

[20] J. A. Recio-García, D. Bridge, B. Díaz-Agudo, and P. A. González-Calero, "CBR for CBR: A Case-Based Template Recommender System for Building Case-Based Systems," in *ECCBR08*, ser. Lecture Notes in Computer Science, K.-D. Althoff, R. Bergmann, M. Minor, and A. Hanft, Eds., vol. 5239. Springer, 2008, pp. 459–473.

[21] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," in *International Joint Conference on Artificial Intelligence*, vol. 14, LAWRENCE ERLBAUM ASSOCIATES LTD. Citeseer, 1995, pp. 1137–1143. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.529

[22] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstei, K. Elmeleegy, and R. Sears, "MapReduce Online," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implemention*, 2010.