Research of Fast Search Algorithm Based on Hadoop Cloud Platform

Fei Guo[*]

Computer College, Civil Aviation Flight University of China, Guanghan, 618307, China

Abstract: Hadoop is a powerful open cloud computing platform, using MapReduce hardled data seg mentation and merging. Personal data in the flat without any protection, could be attacked at any time, as a result, cloud platform on the personal fast search algorithm problem is very important. In this paper, a Hadoop cloud data search platform to have safety rules was proposed, it increase the safety of the cloud data segmentation and merging. The results show that fast search program has really improved cloud platform.

Keywords: Hadoop, fast search algorithm, cloud platform.

1. INTRODUCTION

With rapid development of Internet technology, search system has become the most important filter tool to solve information overload problem, and help users to find the useful content to them from massive data quickly and high effectively. But in practical applications, types of items and number of users may be very large, the generation and accumulation speed of log information in system continue to grow rapidly, and traditional recommendation algorithm used to run on single machine, and limited by its performance, it's not far enough to meet the needs of recommendation computation on mass data [1].

However, MapReduce has also been shown to have significant problems with more complex algorithms, like conjugate gradient, fast Fourier transform and block linear system solver. Moreover, most of these problems use iterative methods to solve them, indicating that MapReduce may not be well suited for algorithms that have an iterative nature. However, there is more than one type of iterative algorithm. To study if MapReduce model is unsuitable for all iterative algorithms or only a certain subset of them, we devised a set of classes for scientific algorithms. Algorithms are divided between these classes by how difficult it is to adapt them to the MapReduce model and their resulting structure. To be able to compare the classes to each other, we selected and adapted algorithms from each class to the MapReduce model and studied their efficiency and scalability. Such a classification allows us to precisely judge which algorithms are more easily adaptable to the MapReduce model and what kind of effect belonging to a specific class has on the parallel efficiency and scalability of the adapted algorithms.

2. WEB ONOTOLOGIES QUERYING

Semantic Web, which is proposed by Tim Berners-Lee, is the vision of next generation of Web. Through combining the concepts of ontology from philosophy into computer science domain, computers can understand the information published in the Semantic Web, and it is possible to exchange semantic information among computers. In the World Wide Web Consortium (W3C) [2] proposed semantic web stack, SPARQL-based Resource Description Framework (RDF) data querying, description logic-based Web Ontology Language (OWL) reasoning, and Semantic Web Rule Language (SWRL)-based OWL ontologism rule reasoning are the core contents of semantic web research.

However, large-scaled ontologism have existed with the explosion of the semantic web technologies, and the amounts of it is rapidly growing ever year. Therefore, these conventional semantic web data querying and reasoning tools do not scale well for large amounts of ontologism because they are designed for use on a single-machine context.

Recent years, cloud computing has become one of the latest research area in both academe and IT industry because of its high-performance and scalability for storing and computing on large-scaled data. Nowadays, Hadoop technologies have become the de-facto standard of Big Data processing. Several researchers have started to combine cloud computing and semantic web technologies to explore high-performance ontology querying and reasoning solutions in the distributed computing context. However, this novel research area is still in the initial stage, lots of key problems need to be solved.

To overcome the drawbacks, this thesis researches on the approaches of distributed querying and reasoning for large-scaled ontology data by utilizing cloud computing technologies. This thesis can establish the theoretical research basis for implementing large-scaled semantic web ontology data management cloud services in the future.

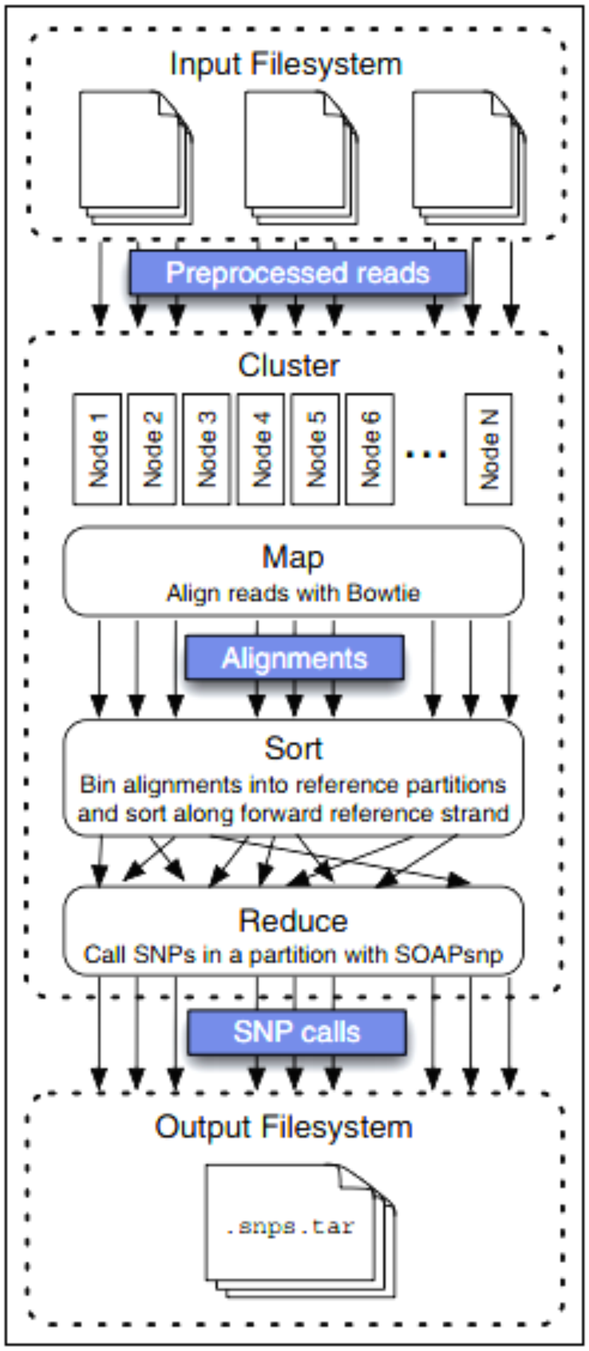

Fig. (1). Workflow.

2.1. Study Process

Workflow is shown in Fig. (**1**) [3]. In order to solve the scalability problems of recommender system, we have proposed several system solutions based on distribute computing open source software framework Apache Hadoop. On the basis of in-depth study of distribute system HDFS and programming ideas MapReduce, in this thesis we have put forward several distributed parallel implementation based on MapReduce programming model to Network-based recommendation algorithm which is proposed these years. On this basis, we have designed and implemented a recommender prototype system based on Hadoop.

Study running mechanism of Hadoop and programming framework MapReduce, combine in-depth analysis of recommender system and recommendation algorithm, especially Network-based recommendation algorithm which is represented by probabilistic spreading algorithm and heat spreading algorithm, and design MapReduce programming implementation solution of Network-based recommendation algorithm on Hadoop. The complex computing tasks are

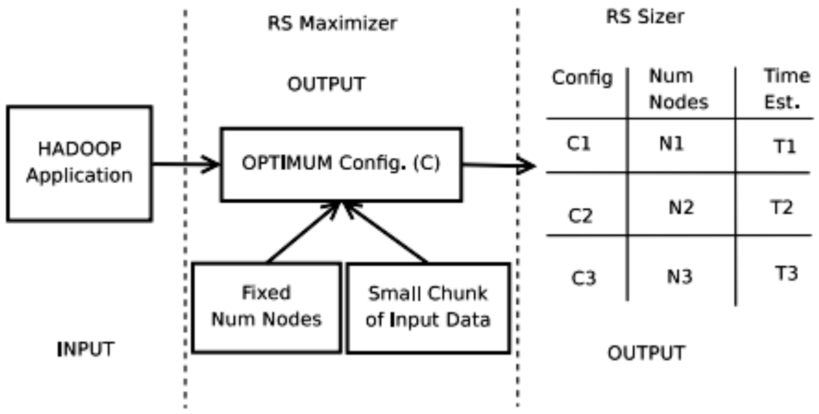

Fig. (2). Overview of the optimization process to provision Hadoop in the cloud.

decomposed into a series of MapReduce job flow for distributed parallel processing on Hadoop and cloud computing platforms; the experimental tests have shown that the algorithm has good parallelism and scalability in cluster.

Based on MapReduce solution for Network-based recommendation algorithm, we have used optimization method such as combiner function and sequence file I/O type, analyzed long tail distribution of dataset and implementation details of algorithm computation process, and proposed improvement MapReduce solutions of Network-based recommendation algorithm on Hadoop by optimization ideas, for instance, Pair and Stripe in the resource allocation matrix computing process, and cut-down to extremely active users.

Study installation, deployment and usage of related open source software include Hadoop, Mahout, Sqoop and Ganglia etc, combine with MapReduce solution of Network-based recommendation algorithm designed in the thesis, after several steps of system requirements, system design for framework and process, system implementation and system testing, design, implement and run a set of recommender prototype system based on Hadoop in cluster environment consisting of multiple computers.

2.2. Method

Hadoop is an open source implementation of the MapReduce programming model. A MapReduce job usually consists of three phases—map, copy and reduce.

The input data is split into chunks of 64MB size (by default). In the map phase, a user dined function operates on every chunk of input data producing intermediate key-value pairs which are stored on local disk. One map process is invoked to process one chunk of input data [4].

In the copy phase, the intermediate key-value pairs are transferred to the location where a reduce process would operate on the intermediate data. In reduce phase, a user dense reduce function operates on the intermediate key value pairs and generates the output. One reduce process is invoked to process a range of keys. Hadoop has over 180 parameters. Examples include number of replicas of input data, number of parallel map/reduce tasks to run, number of parallel connections for transferring data etc. Of these several parameters, this paper focuses on two that nuance the resource utilization in a resource set.

Each map/reduce task runs as a separate process and hence a higher number for these parameters translates into higher parallelization. But too high a value can potentially cause resource contention and degrade overall performance. For example, setting a very high value for this parameter results in large number of simultaneous disk reads results in disk contention. Setting a low value, on the other hand, might under-utilize the resources, and once again reduce performance. Thus, the number of map and reduce tasks per resource set must be chosen such that the resources are maximally utilized, resulting in optimum performance.

Fig. (**2**) and Fig. (**3**) show the time taken by grip to search for a simple rage string in 80 GB of randomly generated data as the number of maps and reduces are varied. Fig. (**3**) shows that: (1) Time taken for grip varies with number of maps, but is independent of number of reduces, (2) The configuration with 8 maps yields the best performance and runs 4× faster when compared to the configuration with 1 map and roughly 1.5× faster when compared to configuration with 24 Maps. Performing rage on input stream of data is inherently computationally intensive and hence grip is CPU dependent. Since each node in the cluster has 8 cores, 8 maps potentially achieve close to optimum CPU utilization. (3) Finally, increasing number of maps from 4 to 8 only marginally improves the performance, whereas changing maps from 1 to 4 almost quadruples the performance. Here, we observe that the disk bandwidth begins to saturate before all the CPUs are fully utilized, giving rise to a small improvement from 4 to 8 maps. Increasing the number of maps further causes disk thrashing as a result of which the performance decreases. Similar results were observed in the 4 node cluster.

$$\chi^2{}_{S^{r1}_{m},S^{r2}_{m}} = \sum_{i=1}^{n} \frac{(S^{r1}_{mi} - S^{r2}_{mi})^2}{(S^{r1}_{mi} + S^{r2}_{mi})} \quad (1)$$

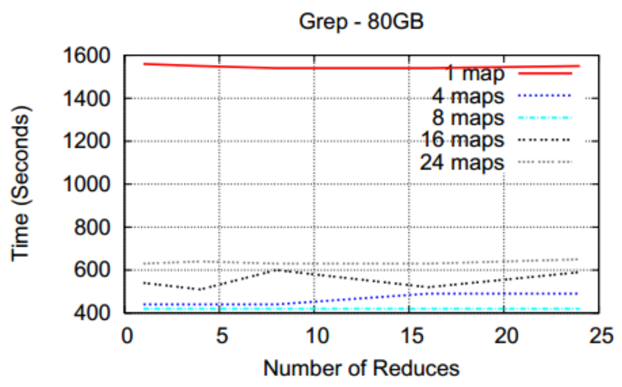

Fig. (3). Time taken to grep a simple string from 80GB of input data. Each line corresponds to a fixed number of maps.

Table 1. Run times of implementation in mapReduce under varying cluster size.

Unknowns	24	500	1000	2000	4000	6000
1 node	259	261	327	278	1938	3810
2 nodes	255	259	298	507	1268	2495
4 nodes	255	236	261	360	721	1374
8 nodes	251	251	291	297	563	824
16 nodes	236	240	278	397	338	511

According to the model, it took 220 s to solve a system with only 24 unknowns in a 16 node cluster, [5] which is definitely very slow for solving a linear system with such a small number of calculations needed. Unfortunately, the tests solving larger systems also showed that the CG MapReduce algorithm does not improve as the size of the data increases. For example, a linear system with 8000 unknowns took almost 2 h to solve using the MapReduce algorithm. These results indicate that most of the time in the MapReduce CG [6] is spent on the background tasks and not on the actual calculations in Table **1**.

3. ANALYSIS OF ALGORITHM

3.1. Algorithm Classes

We have devised a set of classes for scientific algorithms based on how difficult it is to adapt them to the MapReduce model and what steps are required. The algorithms are divided into different classes as follows:

· Algorithms that can be adapted as a single execution of a MapReduce model [7].

· Algorithms that can be adapted as a sequential execution of a constant number of MapReduce models.

· Algorithms where the content of one iteration is represented as an execution of a single MapReduce model.

· Algorithms where the content of one iteration is represented as an execution of multiple MapReduce models.

First class can be considered to represent embarrassingly parallel algorithms and the second class easily parallelizable algorithms. Third and fourth represent iterative algorithms, where some type of synchronization must be performed between each iteration; for example to check the ending condition or to aggregate and broadcast the result of the previous iteration. Algorithms belonging to the fourth class are considered to be more complex iterative algorithms where only some operations in each item ration can be parallelized completely. Algorithms belonging to class 4 are generally difficult to parallelize efficiently, which is even more difficult to achieve when adapting them to the MapReduce model. To study how belonging to a specific class affects the efficiency and scalability of an algorithm, we adapted algorithms from each class to MapReduce and analyzed the results. The algorithms we chose are described in the following chapter.

Apart from belonging to the specific class, the parallel efficiency and scalability is also affected by the individual characteristics of the algorithm. For example, it depends on how large part of the computation stays outside of the MapReduce model [8]. Checking the ending condition in an iterative algorithm or aggregating and processing the final result, when it cannot be done in the reducer method in MapReduce, means that most often some part of the iterative algorithm must be executed outside parallelism, which decreases the parallel efficiency of the whole algorithm. Also,

Table 2. Run times of algorithm.

Objects(thousands)	25	50	100	500	1000	5000
1 node	117	118	125	193	261	819
2 nodes	79	84	89	150	215	756
4 nodes	61	66	72	120	157	316
4868 nodes	52	56	81	114	124	218
16 nodes	44	50	58	99	104	156

algorithms that belong to the second class can become less efficient if the number of sequential MapReduce model executions is large. Switching between different MapReduce models acts as a synchronization step and the input data for each different MapReduce execution must be processed again, meaning there might be no practical difference between the second and third classes when the number of steps in the second is comparable to the number of iterations in the third.

Furthermore, the parallel efficiency does not only depend on how the algorithm was adapted to the MapReduce model or on the inherit characteristics of the algorithm itself, but also on the environment it is executed in. Executing MapReduce applications on different MapReduce frameworks can have a significant impact on the running time of the application, also depending on which class the algorithms used in this application belong to.

3.2. Partitioning Around Medolds

Partitioning Around Medoids (PAM) is an iterative k-medoid clustering algorithm, that has significant value in the data mining domain. The general idea of a k-medoid clustering is that each cluster is represented by its most central element, the medoid, and all comparisons between objects and clusters are reduced into comparisons between objects and the medoids of the clusters.

To cluster a set of objects into k different clusters, the PAM algorithm first chooses k random objects as the initial medoids. As a second step, for each object in the dataset, the distances from each of the k medoids is calculated and the object is assigned to the cluster with the closest medoid. As a result, the dataset is divided into k different clusters. At the next step the PAM algorithm recalculates the medoid positions for each of the clusters, choosing the most central object as the new medoid. This process of dividing the objects into clusters and recalculating the cluster medoid positions is repeated, until there is no change from the previous iteration, meaning the clusters have become stable.

Similar to CG, PAM makes an initial guess of the solution, in this case the clustering, and at each following iteration it improves the accuracy of the solution. Also, as with CG, it is not possible to reduce the whole algorithm to the MapReduce model. However, the content of a whole iteration can be reduced to the MapReduce model, showing that PAM belongs to the third algorithm class. The resulting MapReduce job can be expressed as:

·Map:

-Find the closest medoid and assign the object to its cluster.

-Input: (cluster id, object).

-Output: (new cluster id, object).

·Reduce:

-Find which object is the most central and assign it as a new medoid to the cluster.

-Input: (cluster id, (list of all objects in the cluster)).

-Output: (cluster id, new medoid).

The run times for the integer factorization are given on the Table **2** and speedup is shown on Fig. (**4**). From the Fig. (**4**) it is possible to see that when the factored number is small, there is only a small advantage in using multiple workers in parallel. The speedup is slightly above 1 for 2 node cluster and only reaches 2.22 in 16 node cluster. This is because the number of calculations done was relatively small compared to the background tasks of the framework. However, with the increase of the size of the input number, the speedup started to grow significantly. With the input size of 21 digits, the speedup for two and four node executions was 2.07 and 4.17, showing that there is an ideal gain from using multiple nodes to find the factors when the size of the input is large enough. With a larger number of nodes the speedup does not reach the number of nodes, indicating that calculations were not long enough to get the full benefit from using 16 nodes. The calculated speedup numbers suggest that this algorithm has a good scalability and that algorithms belonging to the first class can be very suitable for the Hadoop MapReduce framework.

4. MASSIVE DATA SEARCH AND PROCESSING BASED ON HADHOOP

With the integration of the mobile network and the Internet, different kinds of data service used by users have become the main way of information transfer. Those service data is transferred over the Internet by the way of IP datagram. At present, the network quality indicators based on NMS can not take control of service effectively according to the characteristics of user behavior or reflect the real user experience of various service. In this case, we need search IP packet continuously, and then study the analysis system of user behavior characteristics, the law of data service, im-

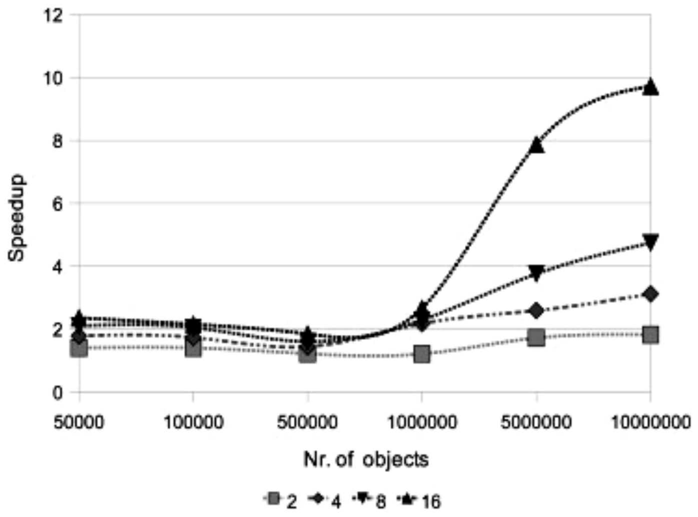

Fig. (4). Parallel speedup for the algorithm with different number of nodes.

prove the predictive ability of the network about the user characteristics and promote the development of future network.

Network packet is the core of this demand and is of great significance to follow-up analysis of data and the characteristics of user behavior. With the beginning of the network data search, massive data rapidly emerges. It is a server test to the resources of database servers. With the rapid increase of data resources, all data analysis and processing job to be completed by a single database system alone can not meet the actual needs. Therefore, we need to enhance capabilities of data processing to meet the data processing requirements of large data environment.

Accuracy of the data analysis can reflect the value of the data and is good for the study of user behavior characteristics. Therefore, the study of the characteristics of the Internet data can help to portray the behavior of the network accurately and give guidance to the practical network deployment and traffic control, promoting the study of service-oriented future Internet architecture and mechanism.

With the rapid development of modern technology, people can query travel knowledge and information through Internet. It impels tourists to prefer to arrange their travel according to their personal willingness. So, self-help travel proportion is increasing year by year. And the tourism information service market becomes booming nowadays. In addition, the sudden and uncertainty of travel process decide that Mobile Internet is the main way that tourists query travel information. Mobile cloud computing introduces the concept of cloud computing into the Mobile Internet. On the one hand, it makes full use of the portability of mobile terminals. On the other hand, it makes up shortages of the mobile terminal in computing and storage capacity. In order to apply mobile cloud computing in the tourism information retrieval service, it's necessary to propose the mobile terminal cloud resources access pattern combining with the characteristics of mobile terminal and travel behavior.

In order to design the mobile terminal cloud resources access pattern based on Hadoop, first of all, it's necessary to deeply understand the basic component of Hadoop, and emphatically analyze the working principle of the two core components: HDFS and MapReduce. The consistency of Hadoop and cloud computing in views and key technologies makes Hadoop become a distributed cloud computing platform. Secondly, the cloud resources access pattern is divided into three sections: data storage strategy, request scheduling algorithm and data response method, and is analyzed under Hadoop common pattern. Then, combining with the mobile terminal characteristics of small computational amount, high concurrency and real-time requirement, the mobile terminal cloud resources access pattern is purposed based on Hadoop. Data storage strategy is composed of tree topology, replication number dynamic optimization model and farthest node selection strategy. The farthest node is got by farthest leaf node algorithm. Localization scheduling algorithm with waiting length threshold is proposed in the respect of request scheduling algorithm. And the response is fed back to users through the HTTP protocol. Finally, it is applied to tourism information retrieval system, and analyzed from the application effect. The results show that the data storage strategy not only can save storage space, but also can improve file system reliability, localization scheduling algorithm greatly increases the data localization probability and reduces the response time, and the response can easily feed back to user by HTTP protocol.

CONCLUSION

In order to deal with huge amounts of data scalability, using a distributed platform to complete social networking service recommendation algorithm is a good choice. Given the inherent mass data storage and processing power of Hadoop, it can effectively solve the difficulties in safe storage and efficient processing, at the same time it can guarantee reliability, effectiveness and security of the data. In this paper, we put forward building social networking service recommendation system on Hadoop cloud platform.

The system is divided into four parts, like data acquisition module, data preprocessing module, data storage module and service recommendation module. In the data acquisition module, we use sina weibo API to access to user data. In the data preprocessing module, FudanNLP is adopted to precede the Chinese word segmentation. In data storage module, we build HBase tables to store data, and use the HBase API to operate the tables. In service recommendation module, we implement the distributed TF-IDF algorithm in the MapReduce model, this algorithm is used to calculate the importance of each word, and to extract the keywords. According to the keywords extracted, you can find the user's interest, and recommend relevant content to the user.

In order to verify the accuracy and validity of the distributed TF-IDF algorithm in this paper, we compare the keywords extracted by the distributed TF-IDF algorithm with the keywords extracted by the Text Rank algorithm for many times. Results show that keywords extracted by these two algorithms are very close, and with the increasing of keywords' number, the results become closer. This proves that the distributed TF-IDF algorithm implemented on MapReduce is accurate and effective. At the same time, due to the distributed TF-IDF algorithm considers the identification problem of keywords, it performs better than the Text Rank algorithm. In addition, compared with Text Rank algorithm of response time, it can be seen that the distributed TF-IDF algorithm has good scalability.

In this paper, the proposed recommendation system based on Hadoop cloud platform has a certain reference value for data mining application in cloud platform, and has certain exploring significance for recommendation system implementation in cloud platform.

CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, pp. 599-616, 2009.

[2] S.N. Srirama, O. Batrashev, and E. Vainikko, "Scicloud: scientific computing on the cloud," In: 10th *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGrid, pp. 579, 2010.

[3] J. Cohen, "Graph twiddling in a MapReduce world," *Computing in Science and Engineering*, vol. 11, pp. 29-41, 2009.

[4] A. Gediminas, and T. Alexander, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, 2005.

[5] S. Schubiger, and B. Hirsbrunner, "A model for software configuration in ubiquitous computing environments", In: *Proceedings of the 1st International Conference on Pervasive Computing,* Zurich, Switzerland, 2002, pp. 181-194.

[6] N. Wang, P. Wang, and B. Zhang, "An improved TF-IDF weights function based on information theory," In: *Proceedings of the International Conference on Computer and Communication Technologies in Agriculture Engineering* (CCTAE), Chengdu, China, 2010, pp. 439-441.

[7] R. O. Duda, P. E. Hart, and D. G. Stork, "*Pattern Classification*," John Wiley & Sons, New York, 2001.

[8] W. Toshihiko, K. Shingo, and F. Ryosuke, "Improvement of collaborative filtering based on fuzzy reasoning model," In: *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, Taipei, 2006, pp. 4790-4795.

Received on: May 26, 2015 Revised on: July 14, 2015 Accepted on: August 10, 2015

© Fei Guo; Licensee *Bentham Open.*

This is an open access article licensed under the terms of the (https://creativecommons.org/licenses/by/4.0/legalcode), which permits unrestricted, non-commercial use, distribution and reproduction in any medium, provided the work is properly cited.