KASR: A Keyword-Aware Service Recommendation Method on MapReduce for Big Data Applications

Shunmei Meng, Wanchun Dou, Xuyun Zhang, Jinjun Chen, *Senior Member, IEEE*

Abstract—Service recommender systems have been shown as valuable tools for providing appropriate recommendations to users. In the last decade, the amount of customers, services and online information has grown rapidly, yielding the big data analysis problem for service recommender systems. Consequently, traditional service recommender systems often suffer from scalability and inefficiency problems when processing or analysing such large-scale data. Moreover, most of existing service recommender systems present the same ratings and rankings of services to different users without considering diverse users' preferences, and therefore fails to meet users' personalized requirements. In this paper, we propose a Keyword-Aware Service Recommendation method, named KASR, to address the above challenges. It aims at presenting a personalized service recommendation list and recommending the most appropriate services to the users effectively. Specifically, keywords are used to indicate users' preferences, and a user-based Collaborative Filtering algorithm is adopted to generate appropriate recommendations. To improve its scalability and efficiency in big data environment, KASR is implemented on Hadoop, a widely-adopted distributed computing platform using the MapReduce parallel processing paradigm. Finally, extensive experiments are conducted on real-world data sets, and results demonstrate that KASR significantly improves the accuracy and scalability of service recommender systems over existing approaches.

Index Terms—recommender system, preference, keyword, Big Data, MapReduce, Hadoop

1 Introduction

1.1 Background

In recent years, the amount of data in our world has been increasing explosively, and analyzing large data sets—so-called "Big Data"— becomes a key basis of competition underpinning new waves of productivity growth, innovation, and consumer surplus [1]. Then, what is "Big Data"?, Big Data refers to datasets whose size is beyond the ability of current technology, method and theory to capture, manage, and process the data within a tolerable elapsed time. Today, Big Data management stands out as a challenge for IT companies. The solution to such a challenge is shifting increasingly from providing hardware to provisioning more manageable software solutions [2]. Big Data also brings new opportunities and critical challenges to industry and academia [3] [4].

Similar to most big data applications, the big data tendancy also poses heavy impacts on service recommender systems. With the growing number of alternative services, effectively recommending services that users preferred has become an important research issue. Service recommender systems have been shown as valuable tools to help users deal with services overload and provide appropriate recommendations to them. Examples of such practical applications include CDs, books, web pages and various other products now use recommender systems [5], [6], [7]. Over the last decade, there has been much research done both in industry and academia on developing new approaches for service recommender systems [8], [9].

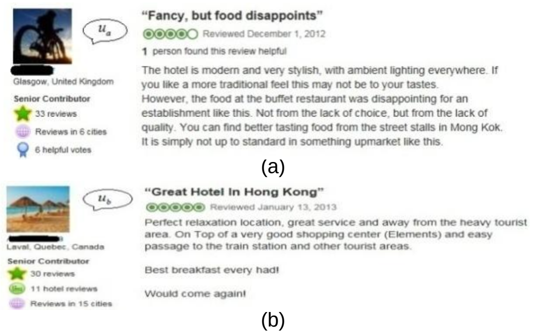
- S. Meng and W. Dou are with the State Key Laboratory for Novel Software Technology, the Dept. of Computer Science and Technology, Nanjing University, China, 210023. E-mail: { shunmei89@gmail.com, douwc@nju.edu.cn}.
- X. Zhang and J. Chen are with the Faculty of Engineering and IT, University of Technology, Sydney, Australia, NSW2007. E-mail: {xyzhanggz, jinjun}@gmail.com.

1.2 Motivation

With the success of the Web 2.0, more and more companies capture large-scale information about their customers, providers, and operations. The rapid growth of the number of customers, services and other online information yields service recommender systems in "Big Data" environment, which poses critical challenges for service recommender systems. Moreover, in most existing service recommender systems, such as hotel reservation systems and restaurant guides, the ratings of services and the service recommendation lists presented to users are the same. They have not considered users' different preferences, without meeting users' personalized requirements. Following is an example in hotel reservation system illustrating such a case.

Example 1. *Alice and Tom are respectively browsing a hotel reservation website to reserve a hotel in Kowloon, Hong Kong. But the ratings and recommendation list of the hotels provided by the website to them are the same. Assuming there are three hotels in Kowloon: A, B and C. Comparing the three hotels, A is convenient to the airport and has a shopping mall nearby; B has convenient transportation with an underground station nearby and owns comfortable accommodation equipment; the breakfast and food of C is delicious and its view is very good. According to the overall ratings provided by the website, B is better than A and A is better than C. However, in this travel, Alice prefers a shopping mall near the hotel and good location, while Tom is concerned about food and wishes good view around the hotel. So hotel B may not be the best choice for them, and hotel A and C may be more* *appropriate to Alice and Tom respectively*.

So the problem is how to provide Alice and Tom a personalized recommendation list respectively to recommend the most appropriate hotels to them in "Big Data" environment efficiently.



[redacted]
Glasgow, United Kingdom
Senior Contributor
33 reviews
Reviews in 6 cities
6 helpful votes

"Fancy, but food disappoints"
Reviewed December 1, 2012
1 person found this review helpful
The hotel is modern and very stylish, with ambient lighting everywhere. If you like a more traditional feel this may not be to your tastes.
However, the food at the buffet restaurant was disappointing for an establishment like this. Not from the lack of choice, but from the lack of quality. You can find better tasting food from the street stalls in Mong Kok. It is simply not up to standard in something upmarket like this.

(a)

[redacted]
Laval, Quebec, Canada
Senior Contributor
30 reviews
11 hotel reviews
Reviews in 15 cities

"Great Hotel In Hong Kong"
Reviewed January 13, 2013
Perfect relaxation location, great service and away from the heavy tourist area. On Top of a very good shopping center (Elements) and easy passage to the train station and other tourist areas.

Best breakfast every had!

Would come again!

(b)

Fig. 1 Two reviews of W Hong Kong hotel.

As shown in Fig.1 (a) and (b), there are two reviews of W Hong Kong hotel snapshotted from a well-known travel review site (www.tripadvisor.com). From Fig.1, we can see that user u_a prefers modern feel and delicious food, while another user u_b is concerned more about good location, great service, breakfast, shopping and convenient transportation. We know that the reviews of the users commented for the hotels are related with both their preferences and the quality of the hotels. So the review of u_b to is useful to both Alice and Tom, while the review of u_a is useful to Tom only, since the review of u_a is irrelevant to the preferences of Alice. Thus it could be found that the information extracted from the reviews of previous users can be used to help recommend appropriate services to new users.

Motivated by these observations, in this paper, we address these challenges through the following contributions: (1) A keyword-aware service recommendation method, named KASR, is proposed in this paper, which is based on a user-based Collaborative Filtering algorithm. (2) In KASR, keywords extracted from reviews of previous users are used to indicate their preferences. Moreover, we implement it on a distributed computing platform, Hadoop, which uses MapReduce as its computing framework.

The remainder of the paper is organized as follows: Section 2 introduces the preliminary knowledge of our method. Then a keyword-aware service recommendation method, named KASR, is described in Section 3. Section 4 presents the implementation of KASR on MapReduce. In Section 5, experiments are designed and analyzed to evaluate the accuracy and scalability of KASR. Related works are presented in Section 6. Section 7 concludes the paper and gives an outlook on possible continuations of our work.

2 Preliminary Knowledge

2.1 Recommender Systems and Collaborative Filtering

Recommender systems developed as an independent research area in the mid-1990s when recommendation problems started focusing on rating models [10], [11]. According to the definition of recommender system in [12], recommender system can be defined as system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful services in a large space of possible options. Current recommendation methods usually can be classified into three main categories: content-based, collaborative, and hybrid recommendation approaches [13]. Content-based approaches recommend services similar to those the user preferred in the past. Collaborative filtering (CF) approaches recommend services to the user that users with similar tastes preferred in the past. Hybrid approaches combine content-based and CF methods in several different ways.

CF algorithm is a classic personalized recommendation algorithm, which is widely used in many commercial recommender systems [13]. In CF based systems, users receive recommendations based on people who have similar tastes and preferences, which can be further classified into item-based CF and user-based CF. In item-based systems, the predicted rating depends on the ratings of other similar items by the same user. While in user-based systems, the prediction of the rating of an item for a user depends upon the ratings of the same item rated by similar users. And in this work, we will take advantage of a user-based CF algorithm to deal with our problem. More details of user-based CF algorithm can be found in Appendix A.1.

2.2 Cloud Computing and MapReduce

Cloud computing is a successful paradigm of service oriented computing and has revolutionized the way computing infrastructure is abstracted and used. The major goal of cloud computing is to share resources, such as infrastructure, platform, software, and business process [14].

Cloud computing can provide effective platforms to facilitate parallel computing, which has gained significant attention in recent years to process large volume of data. There are several cloud computing tools available, such as Hadoop (http://hadoop.apache.org/), Mahout (http://mahout.apache.org/), MapReduce of Google [15], the Dynamo of Amazon.com [16], the Dryad of Microsoft and Neptune of Ask.com [17], etc. Among these tools, Hadoop is the most popular open source cloud computing platform inspired by MapReduce and Google File System papers [18], which supports MapReduce programming framework and mass data storage with good fault tolerance. MapReduce is a popular distributed implementation model proposed by Google, which is inspired by map and reduce operations in the Lisp programming language. More details about MapReduce can be found in Appendix A.2.

Nowadays, the trend "everything as a service" has been creating a Big Services era due to the foundational architecture of services computing. And "servicelization" is the way of offering social networking services, big data analytics, and Internet services [19] [20]. Thus the cloud computing tools aforementioned can be used to improve the scalability and efficiency of service recommendation methods in the "Big Data" environment.

3 Keyword-Aware Service Recommendation Method

Definition 1 (Keyword-Aware Service Recommendation

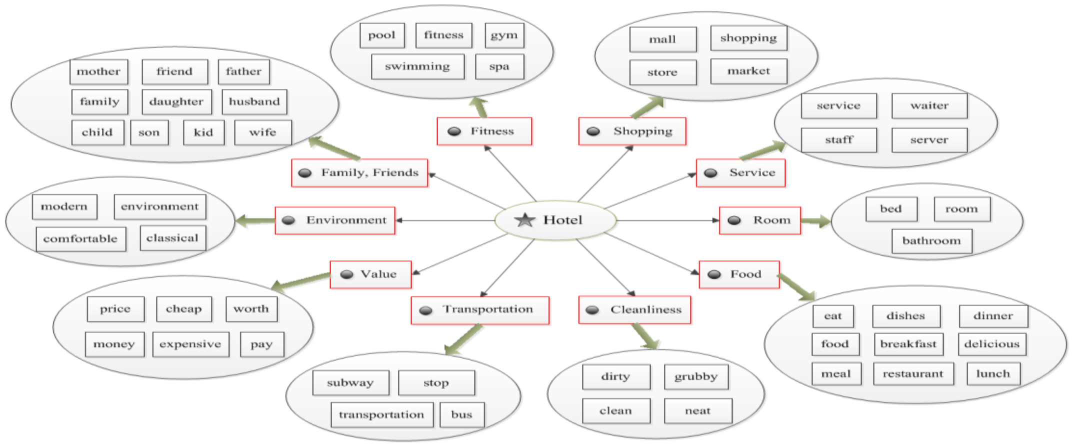


Fig. 2 An example of a simple domain thesaurus of hotel reservation system.

method, KASR). In this paper, we propose a keyword-aware service recommendation method, named KASR. In this method, keywords are used to indicate both of users' preferences and the quality of candidate services. A user-based CF algorithm is adopted to generate appropriate recommendations. KASR aims at calculating a personalized rating of each candidate service for a user, and then presenting a personalized service recommendation list and recommending the most appropriate services to him/her.

Moreover, to improve the scalability and efficiency of our recommendation method in "Big Data" environment, we implement it in a MapReduce framework on Hadoop by splitting the proposed algorithm into multiple MapReduce phases. Table 1 summarizes the basic symbols and notations used in this paper.

TABLE 1
Basic symbols and notations

Symbol	Definition
K	The keyword-candidate list, $K=\{k_1, k_2, \ldots, k_n\}$
APK	The preference keyword set of the active user
PPK	The preference keyword set of a previous user
$sim(APK,PPK)$	The similarity between APK and PPK
$\vec{W}_P$	A preference weight vector
$\vec{W}_{AP}$	The preference weight vector of the active user
$\vec{W}_{PP}$	The preference weight vector of a previous user

3.1 Keyword-candidate List and Domain Thesaurus

In our method, two data structures, "keyword-candidate list" and "specialized domain thesaurus", are introduced to help obtain users' preferences.

Definition 2 (Keyword-candidate list). The keyword-candidate list is a set of keywords about users' preferences and multi-criteria of the candidate services, which can be denoted as $K = \{k_1, k_2, ..., k_n\}$, n is the number of the keywords in the keyword-candidate list. An example of a simple keyword-candidate list of the hotel reservation system is described in Table 2. Keywords in the keyword-candidate list can be a word or multiple words related with the quality criteria of candidate services.

In this paper, the preferences of previous users will be extracted from their reviews for candidate services and formailized into a keyword set. Usually, since some of words in reviews can not exactly matach the corresponding keywords in the keyword-candidate list which characterize the same aspects as the words. The corresponding keywords should be extracted as well. In this paper, we assume that specialized domain thesauruses are built to support the keyword extraction, and different domain thesauruses are built for different service domains.

TABLE 2
Keyword-candidate list of hotel reservation system

No.	Keyword	No.	Keyword	No.	Keyword
1	Service	7	Transportation	13	Airport; Train
2	Room	8	Family; Friends	14	Wi-Fi
3	Shopping	9	Location	15	Environment
4	Cleanliness	10	View	16	Bar
5	Food	11	Quite	17	Beach
6	Value	12	Fitness		

Definition 3 (Domain thesaurus). A domain thesaurus is a reference work of the keyword-candidate list that lists words grouped together according to the similarity of keyword meaning, including related and contrasting words and antonyms [21] [22]. An example of a simple domain thesaurus of hotel reservation system is shown in Fig. 2. As shown in Fig. 2, the words in the red rectangle are the keywords in the corresponding keyword-candidate list, and the words in the ovals are the related words of the keywords. Often, domain thesauruses are updated regularly to ensure the timeliness of the words.

3.2 A Keyword-aware Service Recommendation Method

The main steps of KASR are depicted in Fig. 3, which are described in detail as follows.

(1) Capture user preferences by a keyword-aware approach:

In this step, the preferences of active users and previous users are formalized into their corresponding preference keyword sets respectively. In this paper, an active user refers to a current user needs recommendation.

- *Preferences of an active user.* An active user can give his/her preferences about candidate services by selecting keywords from a keyword-candidate list, which reflect the quality criteria of the services he/she is concerned about. The preference keyword set of the active user can be de-

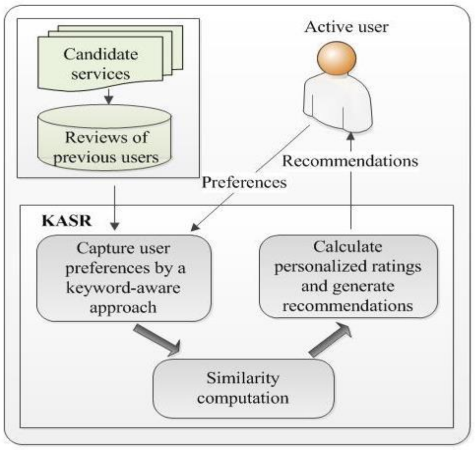

Candidate services
Reviews of previous users
Active user
Recommendations
Preferences
KASR
Capture user preferences by a keyword-aware approach
Calculate personalized ratings and generate recommendations
Similarity computation

Fig. 3 KASR's main steps.

noted as $APK = \{ak_1, ak_2, ..., ak_l\}$, where ak_i $(1 \le i \le l)$ is the i^{th} keyword selected from the keyword-candidate list by the active user, l is the number of selected keywords.

Besides, the active user should also select the importance degree of the keywords. The importance degree of the keywords is shown in Table 3: "1" represents the general, "3" represents important and "5" represents very important.

TABLE 3
Importance degree of the keywords

Measurement	general		important		very important
Importance degree	1	2	3	4	5

- *Preferences of previous users.* The preferences of a previous user for a candidate service are extracted from his/her reviews for the service according to the keyword-candidate list and domain thesaurus. And a review of the previous user will be formalized into the preference keyword set of him/her, which can be denoted as $PPK = \{pk_1, pk_2, \ldots, pk_h\}$, where pk_i $(1 \le i \le h)$ is the i^{th} keyword extracted from the review, h is the number of extracted keywords.

The keyword extraction process is described as follows:

a) Preprocess: Firstly, HTML tags and stop words in the reviews snippet collection should be removed to avoid affecting the quality of the keyword extraction in the next stage. And the Porter Stemmer algorithm (keyword stripping) [23] is used to remove the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up Information Retrieval systems [23].

b) Keyword extraction: In this phase, each review will be transformed into a corresponding keyword set according to the keyword-candidate list and domain thesaurus. If the review contains a word in the domain thesaurus, then the corresponding keyword should be extracted into the preference keyword set of the user. For example, if a review of a previous user for a hotel has the word "spa", which is corresponding to the keyword ``Fitness'' in the domain thesaurus, then the keyword ``Fitness'' should be contained in the preference keyword set of the previous user. If a keyword appears more than once in a review, the times of repetitions will be recorded. In this paper, it is regarded that keywords appearing multiple times are more important. The times of repetitions will be used to calculate the weight of the keyword in preference keyword set in the next step.

(2) Similarity computation:
The second step is to identify the reviews of previous users who have similar tastes to an active user by finding neighborhoods of the active user based on the similarity of their preferences. Before similarity computation, the reviews unrelated to the active user's preferences will be filtered out by the intersection concept in set theory. If the intersection of the preference keyword sets of the active user and a previous user is an empty set, then the preference keyword set of the previous user will be filtered out.

Two similarity computation methods are introduced in our recommendation method: an approximate similarity computation method and an exact similarity computation method. The approximate similarity computation method is for the case that the weights of the keywords in the preference keyword set are unavailable, while the exact similarity computation method is for the case that the weight of the keywords are available.

a) Approximate similarity computation
A frequently used method for comparing the similarity and diversity of sample sets, Jaccard coefficient, is applied in the approximate similarity computation.

Jaccard coefficient is measurement of asymmetric information on binary (and non-binary) variables, and it is useful when negative values give no information. The similarity between the preferences of the active user and a previous user based on Jaccard coefficient is described as follows:

$$sim(APK, PPK) = Jaccard(APK, PPK) = \frac{|APK \cap PPK|}{|APK \cup PPK|} \tag{1}$$

where APK is the preference keyword set of the active user, PPK is the preference keyword set of a previous user. And the weight of the keywords is not considered in this approach.

Algorithm 1, SIM-ASC, illustrates the functionality of the approximate similarity computation method.

```
Algorithm 1:  SIM-ASC (Approximate Similarity Computation )
Input:  The preference keyword set of the active user $APK$
        The preference keyword set of a previous user $PPK_j$
Output:  The similarity of $APK$ and $PPK_j$,  $sim_{ASC}(APK, PPK_j)$
1:  $sim_{ASC}(APK, PPK_j) = \frac{|APK \cap PPK_j|}{|APK \cup PPK_j|}$
2: return the similarity of $APK$ and $PPK_j$,  $sim_{ASC}(APK, UPK_j)$
```

b) Exact similarity computation
A cosine-based approach is applied in the exact similarity computation, which is similar to the Vector Space Model (VSM) in information retrieval [24] [25].

Definition 4 (Preference weight vector). In this cosine-based approach, The preference keyword sets of the active user and previous users will be transformed into n-dimensional weight vectors respectively, namely preference weight vector, which can be denoted as $\vec{W_P} = [w_1, w_2, \ldots, w_n]$, n is the number of keywords in the keyword-candidate list, w_i is the weight of the keyword

k_i in the keyword-candidate list. If the keyword k_i is not contained in the preference keyword set, then the weight of k_i in the preference weight vector is 0, i.e., w_i=0. The preference weight vectors of the active user and a previous user are noted as $\vec{W}_{AP}$ and $\vec{W}_{PP}$, respectively.

In this paper, we use the Analytic Hierarchy Process (AHP) model to decide the weight of the keywords in the preference keyword set of the active user. AHP method is provided by Saaty in 1970s to choose the best satisfied business role for its hierarchy nature [26]. The weight computing based on the AHP model is decied as follows:

Firstly, we construct the pair-wise comparison matrix in terms of the relative importance between each two keywords. The pair-wise comparison matrix $A_m = (a_{ij})_m$ must satisfy the following properties, a_{ij} represents the relative importance of two keywords:

1) $a_{ij} = 1, \quad i = j = 1,2,3,..., m$

2) $a_{ij} = 1/a_{ji}, \quad i, j = 1,2,3,..., m \text{ and } i \neq j$

3) $a_{ij} = a_{ik}/a_{jk}, \quad i, j, k = 1,2,3,..., m \text{ and } i \neq j$

After checking the consistence of the matrix, then we calculate the weight by the following function:

$$w_i = \frac{1}{m}\sum_{j=1}^{m}\frac{a_{ij}}{\sum_{k=1}^{m}a_{kj}} \tag{2}$$

where a_{ij} is the relative importance between two keywords, m is the number of the keywords in the preference keyword set of the active user.

The weight vector of the preference keyword set of a previous user can be decided by *the term frequency/inverse document frequency* (*TF-IDF*) measure [27], which is one of the best-known measures for specifying the weight of keywords in Information Retrieval.

In the *TF-IDF* approach, to calculate the preference weight vector of a previous user u', "all reviews" by user u' should be collected. Here, "all reviews" contain the reviews by user u' for the candidate services and similar services not in the candidate services. The reviews should also be transformed into keyword sets respectively according to the keyword-candidate list and the domain thesaurus.

TF, the term frequency of the keyword pk_i in the preference keyword set of user u' is defined as

$$TF = \frac{N_{pk_i}}{\sum_g N_{pk_i}} \tag{3}$$

where N_{pk_i} is the number of occurrences of the keyword pk_i in all the keyword sets of the reviews commented by the same user u', g is the number of the keywords in the preference keyword set of the user u'.

The inverse document frequency (*IDF*) is obtained by dividing the number of all reviews by the number of reviews containing the keyword pk_i.

$$IDF = \log\frac{|R'|}{|r': pk_i \in r'|} \tag{4}$$

where $|R'|$ is the total number of the reviews commented by user u', and $|r': pk_i \in r'|$ is the number of reviews where keyword pk_i appears. So the *TF-IDF* weight of the keyword pk_i in the preference keyword set of user u' can be decided by the following function:

$$w_{pk_i} = TF \times IDF = \frac{N_{pk_i}}{\sum_g N_{pk_i}} \times \log\frac{|R'|}{|r': pk_i \in r'|} \tag{5}$$

Then the similarity based on the cosine-based approach is defined as follows:

$$sim(APK, PPK) = \cos(\vec{W}_{AP}, \vec{W}_{PP}) = \frac{\vec{W}_{AP} \bullet \vec{W}_{PP}}{\|\vec{W}_{AP}\|_2 \times \|\vec{W}_{PP}\|_2}$$
$$= \frac{\sum_{i=1}^{n}\vec{W}_{AP,i} \times \vec{W}_{PP,i}}{\sqrt{\sum_{i=1}^{n}\vec{W}^2_{AP,i}}\sqrt{\sum_{i=1}^{n}\vec{W}^2_{PP,i}}} \tag{6}$$

where $\vec{W}_{AP}$ and $\vec{W}_{PP}$ are respectively the preference weight vectors of the active user and a previous user. $\vec{W}_{AP,i}$ is the i-th dimension of $\vec{W}_{AP}$ and represents the weight of the keyword k_i in preference keyword set *APK*, $\vec{W}_{PP,i}$ is the i-th dimension of $\vec{W}_{PP}$ and represents the weight of the keyword k_i in preference keyword set *PPK*.

Algorithm 2, SIM-ESC, illustrates the functionality of the exact similarity computation method.

Algorithm 2: SIM-ESC (Exact Similarity Computation)

```
Input: The preference keyword set of the active user APK
       The preference keyword set of a previous user $PPK_j$
Output: The similarity of APK and $PPK_j$, $sim_{ESC}(APK, PPK_j)$
1: for each keyword $k_i$ in the keyword-candidate list
2:   if $k_i \in APK$ then
3:     get_$\vec{W}_{AP,i}$ by formula (2)
4: else $\vec{W}_{AP,i} = 0$
5: end if
6:   if $k_i \in PPK_j$ then
7:     get_$\vec{W}_{PP_j,i}$ by formula (5)
8: else $\vec{W}_{PP_j,i} = 0$
9:   end if
10: end for
11: get $sim_{ESC}(APK, UPK_j)$ by formula (6)
12: return the similarity of APK and $PPK_j$, $sim_{ESC}(APK, UPK_j)$
```

(3) Calculate personalized ratings and generate recommendations:

Based on the similarity of the active user and previous users, further filtering will be conducted. Given a threshold δ, if $sim(APK, PPK_j) < \delta$, the preference keyword set of a previous user PPK_j will be filtered out, otherwise PPK_j will be retained. The thresholds given in two similarity computation methods are different, which are both empirical values.

Once the set of most similar users are found, the personalized ratings of each candidate service for the active user can be calculated. Finally, a personalized service recommendation list will be presented to the user and the service(s) with the highest rating(s) will be recommended to him/her.

Here, we use a weighted average approach to calculate the personalized rating *pr* of a service for the active user.

$$pr = \bar{r} + k\sum_{PPK_j \in \hat{R}} sim(APK, PPK_j) \times (r_j - \bar{r}) \qquad (7)$$

$$k = 1 \Big/ \sum_{PPK_j \in \hat{R}} sim(APK, PPK_j)$$

where $sim(APK, PPK_j)$ is the similarity of the preference keyword set of the active user APK and the preference keyword set of a previous user PPK_j; multiplier k serves as a normalizing factor; $\hat{R}$ denotes the set of the remaining preference keyword sets of previous users after filtering; r_j is the rating of the corresponding review of PPK_j, and $\bar{r}$ is defined as the average ratings of the candidate service.

Repeating the steps above, we can calculate the personalized ratings of all candidate services for the active user. Then we can rank the services by the personalized ratings and present a personalized service recommendation list to him/her. Without loss of generality, we assume that the services with higher ratings are more preferable to the user. So the service(s) with the highest rating(s) will be recommended to the active user. Alternatively, we can recommend the Top-K services to the user.

Algorithm 3: Basic Algorithm of KASR

```
Input: The preference keyword set of the active user APK
       The candidate services WS={ws_1, ws_2, …, ws_N}
       The threshold $\delta$ in the filtering phase
       The number K
Output: The services with the Top-K highest ratings {tws_1, tws_2, …, tws_K}
1: for each service $ws_i \in WS$
2:    $\hat{R} = \Phi, sum = 0, r = 0$
3:    for each review $R_j$ of service $ws_i$
4:       process the review into a preference keyword set $PPK_j$
5:       if $PPK_j \cap APK \neq \Phi$ then
6:          insert $PPK_j$ into $\hat{R}$
7:       end if
8:    end for
9:    for each keyword set $PPK_j \in \hat{R}$
10:      $sim(APK, PPK_j) = SIM(APK, PPK_j)$
   //SIM(APK, PPK_j) can be SIM-ASC(APK, PPK_j) or SIM-ESC(APK, PPK_j)
11:      if $sim(APK, PPK_j) < \delta$ then
12:         remove $PPK_j$ from $\hat{R}$
13:      else $sum=sum+1, r=r+r_j$
14:      end if
15:   end for
16:   $\bar{r} = r / sum$
17:   get $pr_i$ by formula (7)
18: end for
19: sort the services according to the personalized ratings $pr_i$
20: return the services with the Top-K highest ratings
    {tws_1, tws_2, …, tws_K}
```

Algorithm 3 illustrates the basic algorithm of KASR. The input contains the preference keyword set of the active user *APK*, the candidate services $WS = \{ws_1, ws_2, ..., ws_N\}$, the threshold δ in the filtering phase, and the number K. In line 2, $\hat{R}$ is used to store the remaining preference keyword sets of previous users, and *sum* is to record the number of the remaining preference keyword sets of previous users. Line 3 to line 8 is used to process each review of the previous users into the corresponding preference keyword sets, and then do a simple filtering to filter out the reviews unrelated with the active user's preferences. Line 9 to line 15 are to calculate the similarity of *APK* and PPK_j, and then filter out the keyword set PPK_j whose similarity with *APK* is less than the threshold δ. In this paper, there are two methods to calculate the similarity: approximate similarity computation (see Algorithm 1) and exact similarity computation (see Algorithm 2). Line 16 to line 18 is to calculate the personalized ratings of the candidate services for the active user. Finally, line 19 and line 20 is to sort the candidate services according to the personalized ratings and recommend the services with the Top-K highest ratings to the active user.

For convenience, KASR with the approximate and exact similarity computation methods are denoted as KASR-ASC and KASR-ESC, respectively. Suppose there are *N* candidate services and each service with *R* reviews on average. Moreover, suppose that there are *n* keywords in the keyword-candidate list. Then, the time complexity of KASR-ASC and KASR-ESC are *O*(*NR*) and *O*(*NRn*), respectively.

4 Implementation on MapReduce

To improve the scalability and efficiency of KASR in "Big Data" environment, we implement it in a MapReduce framework on Hadoop platform. The whole computation flowcharts of KASR-ASC and KASR-ESC on MapReduce are respectively shown in Fig. 4 (a) and Fig. 4 (b). Due to the space limit, the notations of symbols in Fig. 4 (a) and Fig. 4 (b) are explained in detail in Appendix B.1 and Appendix B.2, respectively.

(1) KASR-ASC on MapReduce

Fig. 4 (a) shows the computation flowchart of KASR-ASC on MapReduce, which consists of three steps. And Step 1 is offline executed, Step 2 and Step 3 are online executed.

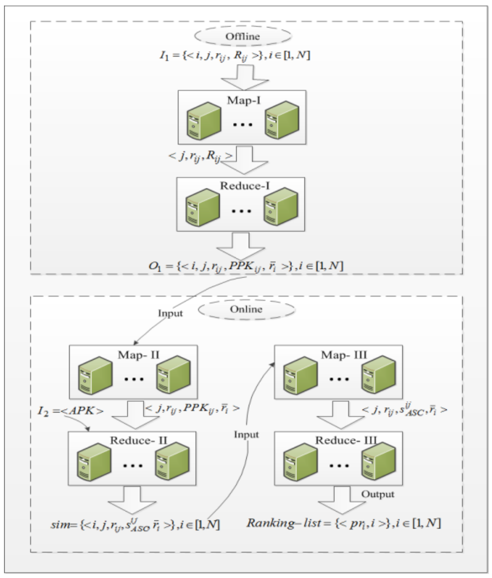
Step 1: The first step is to process the reviews for candidate services by previous users into their preference keyword sets and compute the average ratings for each candidate service. Map-I: Map $< i, j, r_{ij}, R_{ij} >$ on *i* such that the tuples with the same *i* are shuffled to the same node in the form of $< j, r_{ij}, R_{ij} >$. Reduce-I: Take $< j, r_{ij}, R_{ij} >$ as the input and emit $< i, j, r_{ij}, PPK_{ij}, \bar{r}_i >$ for each input of Map-I. The output of Reduce-I $O_1 = \{< i, j, r_{ij}, PPK_{ij}, \bar{r}_i >\}, i \in [1, N]$ will be used as the input of Map-II to calculate the similarity.

Step 2: The second step is to compute the similarity between the active user and previous users. Map-II: Map $< i, j, r_{ij}, PPK_{ij}, \bar{r}_j >$ on *i*, and tuples with the same *i* are shuffled to the same node in form of $< j, r_{ij}, PPK_{ij}, \bar{r}_i >$. Reduce-II: Take <*APK*> and $< j, r_{ij}, PPK_{ij}, \bar{r}_i >$ as the input, then emit $sim = < i, j, r_{ij}, s^{ij}_{ASC}, \bar{r}_i >, i \in [1, N]$.

Step 3: The third step aims to calculate the personalized rating of each candidate service and present a personalized recommendation list to the active user. Based on the output of this step, the recommendation can be obtained. Map-III: Map $< i, j, r_{ij}, s^{ij}_{ASC}, \bar{r}_i >$ on *i* so that the tuples with the same *i* are shuffled to the same node in form of $< j, r_{ij}, s^{ij}_{ASC}, \bar{r}_i >$. Reduce-III: Take $< j, r_{ij}, s^{ij}_{ASC}, \bar{r}_i >$ as the input, and emit $Ranking\text{–}list = \{< pr_i, i >\}, i \in [1, N]$, where pr_i is the personalized rating of the active user to service *i*. The tuples of the output are ordered by the services id *i*, which is just the personalized service recommendation list to the active user.

(2) KASR-ESC on MapReduce

Fig. 4 (b) shows the computation flowchart of KASR-ESC


(a)

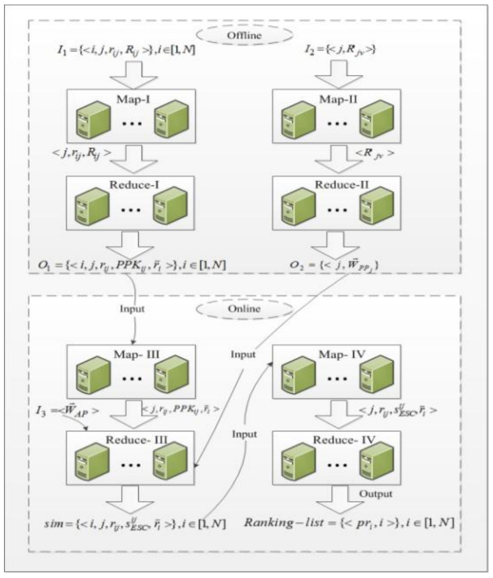

(b)

Fig. 4 The computation flowchart of KASR on MapReduce. (a) shows the computation flowchart of KASR-ASC on MapReduce, which consists of three steps; (b) shows the computation flowchart of KASR-ESC on MapReduce, which consists of four steps.

on MapReduce, which consists of four steps. Step 1 and Step 2 are offline executed, and Step 3 and Step 4 are online executed.

Step 1: The first step of flowchart of KASR-ESC on MapReduce is the same as Step 1 of the flowchart of KASR-ASC on MapReduce.

Step 2: The second step is to process all reviews of each previous user into the corresponding keyword sets respectively and takes advantage of the *TF-IDF* measurement to calculate the preference weight vectors of the previous users. Map-II: Map $< j, R'_{jv} >$ on j such that reviews of the same user j are shuffled to the same node in form of $< R'_{jv} >$. Reduce-II: Take $< R'_{jv} >$ as the input and emit $< j, \vec{W}_{PP_j} >$. $\vec{W}_{PP_j}$ is the preference weight vector of a previous user j. And the tuples $\{< j, \vec{W}_{PP_j} >\}$ will be used to calculate the similarity in Reduce-III.

Step 3: The third step is to compute the similarity between the active user and previous users. Map-III: Map $< i, j, r_{ij}, PPK_{ij}, \bar{r}_i >$ on i, and tuples with the same i are shuffled to the same node in form of $< j, r_{ij}, PPK_{ij}, \bar{r}_i >$. Reduce-III: Take $< \vec{W}_{AP} >$, $< j, \vec{W}_{PP_j} >$ and $< j, r_{ij}, PPK_{ij}, \bar{r}_i >$ as the input, then emit $sim =< i, j, r_{ij}, s^{ij}_{ESC}, \bar{r}_i >, i \in [1, N]$.

Step 4: The last step is the same as Step 3 of the flowchart of KASR-ASC on MapReduce. Based on the output of this step, we can present a personalized service recommendation list to the active user and recommend the most appropriate services to him/her.

5 Experimental Evaluation

In this section, experiments are designed and analysed to evaluate the accuracy and scalability of KASR. To evaluate the performance of KASR in accuracy, we compare KASR with other two well-known recommendation methods: user-based algorithm using Pearson Correlation Coefficient (PCC) and item-based algorithm using PCC, which are called as UPCC [13] and IPCC [28] respectively. Three metrics are used to evaluate the accuracy: Mean Absolute Error (MAE) [29], Mean Average Precision (MAP) [30] and Discounted Cumulative Gain (DCG) [31]. The definitions of MAE, MAP and DCG can be found in Appendix C.1. As to the scalability, a well-accepted scalability metric, Speedup [32], is adopted to measure the performance in the scalability of KASR.

5.1 Experiment Setup and Datasets

Technically, our experiments are conducted in a Hadoop platform. And to evaluate the accuracy and scalability of KASR, two kinds of dataset are adopted in the experiments: a real dataset and a synthetic dataset. Due to the space limit, more details about the experiment settings and dataset can be found in Appendix C.2 and Appendix C.3, respectively.

5.2 Experiment Evaluation

Two groups of experiments are conducted to evaluate the accuracy and scalability of KASR. In the first one, we compare KASR with UPCC and IPCC in MAE, MAP and DCG to evaluate the accuracy of KASR. The other is to explore the scalability of KASR.

5.2.1 Accuracy evaluation

(1) Comparison of UPCC, IPCC, KASR-ASC and KASR-ESC in MAE

MAE is a statistical accuracy metric often used in CF methods to measure the prediction quality. And the Normalized Mean Absolute Error (NMAE) metric is also used to measure the prediction accuracy. The lower the MAE or NMAE presents the more accurate predictions.

Fig. 5 shows the MAE and NMAE values of UPCC, IPCC, KASR-ASC and KASR-ESC. It could be found that the MAE and NMAE vaules of KASR-ASC and KASR-ESC are much lower than UPCC and IPCC (e.g., the MAE and NMAE values of KASR-ASC are respectively 30.02% ((0.6792-0.4753) / 0.6792 = 30.02%) and 35.73% lower than UPCC. And the MAE and NMAE values of KASR-ESC are respectively 23.28% and 22.87% lower than IPCC). Thus our methods KASR-ASC and KASR-ESC can provide more accurate predictions than traditional methods UPCC and IPCC.

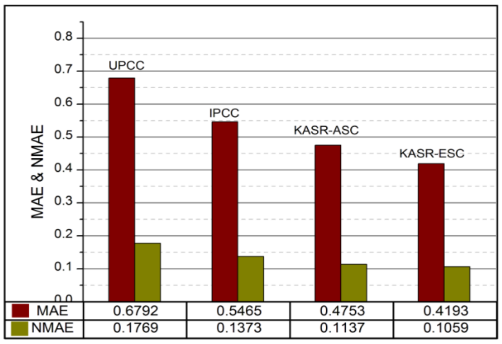
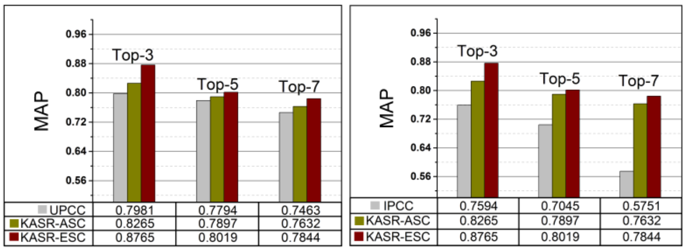


Fig. 5 Comparison of UPCC, IPCC, KASR-ASC and KASR-ESC in MAE.

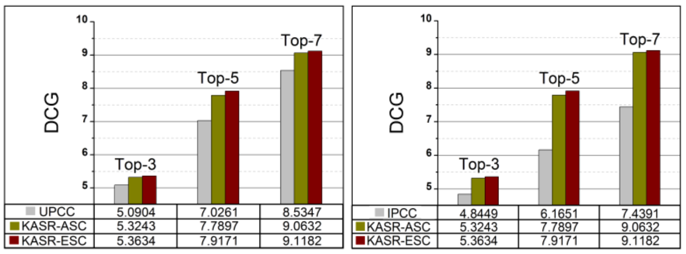
(2) Comparison of UPCC, IPCC, KASR-ASC and KASR-ESC in MAP and DCG

In most service recommender systems, users tend to be recommended the top services of the returned result list. The services in higher position, especially the first position, should be more satisfying than the services in lower position of the returned result list. To evaluate the quality of Top-K service recommendation list, MAP and DCG are used as performance evaluation metrics. And the higher MAP or DCG presents the higher quality of the predicted service recommendation list.



(a) (b)

Fig.6 Comparison of UPCC, IPCC, KASR-ASC and KASR-ESC in the MAP values of Top-K (K=3, 5, 7) recommendation list. (a) shows the comparison of UPCC, KASR-ASC and KASR-ESC in MAP. (b) shows the comparison of IPCC, KASR-ASC and KASR-ESC in MAP.



(a) (b)

Fig.7 Comparison of UPCC, IPCC, KASR-ASC and KASR-ESC in the DCG values of Top-K (K=3, 5, 7) recommendation list. (a) shows the comparison of UPCC, KASR-ASC and KASR-ESC in DCG. (b) shows the comparison of IPCC, KASR-ASC and KASR-ESC in DCG.

Fig. 6 and Fig. 7 respectively show the MAP values and DCG values of Top-K (K=3, 5, 7) recommendation list of UPCC, IPCC, KASR-ASC and KASR-ESC. From Fig. 6 and Fig.7, we can see that the MAP values and DCG values of KASR-ASC and KASR-ESC are comparatively higher than UPCC and IPCC. It also could be found that the MAP values decrease when K increases, while the DCG values increase when K increases.

For a more intuitive illustration, the percentage values that KASR-ASC and KASR-ESC outperform UPCC and IPCC in MAP and DCG of Top-K (K=3, 5, 7) recommendation list are calculated and listed inTabel 4 and Table 5. Table 4 (a) presents the percentage values of KASR-ASC and KASR-ESC outperform UPCC in MAP of Top-K recommendation list (e.g. When K=3, KASR-ASC outperforms UPCC 3.56% ((0.8265-0.7981)/0.7981=3.56%) in MAP of Top-3 recommendation list). Table 4 (b) presents presents the percentage values of KASR-ASC and KASR-ESC outperform IPCC in MAP of Top-K recommendation list. (a) and (b) in Table 5 respectively present the percentage values of KASR-ASC and KASR-ESC outperform UPCC and IPCC in DCG of Top-K (K=3, 5, 7) recommendation list. It could be found that KASR-ASC and KASR-ESC can provide more accurate service recommendation list than UPCC and IPCC.

TABLE 4
The percentage values that KASR-ASC and KASR-ESC outperform UPCC and IPCC in MAP of Top-K (K=3, 5, 7) recommendation list

MAP	Top-3	Top-5	Top-7
KASR-ASC/UPCC	3.56%	1.32%	2.26%
KASR-ESC/UPCC	9.82%	2.89%	5.11%

(a)

MAP	Top-3	Top-5	Top-7
KASR-ASC/IPCC	8.84%	10.86%	32.71%
KASR-ESC/IPCC	15.42%	13.83%	36.39%

(b)

TABLE 5
The percentage values that KASR-ASC and KASR-ESC outperform UPCC and IPCC in DCG of Top-K (K=3, 5, 7) recommendation list

DCG	Top-3	Top-5	Top-7
KASR-ASC/UPCC	4.59%	10.87%	6.19%
KASR-ESC/UPCC	5.36%	12.68%	6.84%

(a)

DCG	Top-3	Top-5	Top-7
KASR-ASC/IPCC	9.89%	26.35%	21.83%
KASR-ESC/IPCC	10.70%	28.41%	22.57%

(b)

Generally speaking, KASR-ASC and KASR-ESC perform better than traditional methods, UPCC and IPCC, in MAE, MAP and DCG. Thus the personalized service recommendation lists provided by our method would satisfy users better.

5.2.2 Scalability evaluation

A well-accepted scalability metric, Speedup [33], is adopted to measure the performance in the scalability of KASR. Speedup refers to how much a parallel algorithm is faster than a corresponding sequential algorithm, which can be defined as follows:

$$S_p = \frac{T_1}{T_p} \tag{8}$$

where p is the number of processors, T_1 is the sequential execution time, T_p is the parallel execution time with p processors. If the speedup has a linear relation with the numbers of nodes with the datasize fixed, the algorithm will have good scalability.

To verify the scalability of KASR, experiment is conducted respectively in a cluster of nodes ranging from 1 to 8. There are 4 synthetic datasets used in the experiments (128M, 256M, 512M and 1G datasize). Fig. 8 shows the speedup of KASR (Here, KASR-ESC method is adopted in the scalability experiment). From Fig. 8, we can see that the speedup of KASR increases relative linearly with the growth of the number of nodes. Meanwhile, larger dataset obtained a better speedup. When the datasize is 1G and the number of nodes is 8, the speedup value reaches 6.412, which is 80.15% (6.412/8=80.15%) of the idealspeedup. The experimental result shows that KASR on Map-Reduce in Hadoop platform has good scalability over “Big Data” and performs better with larger dataset.

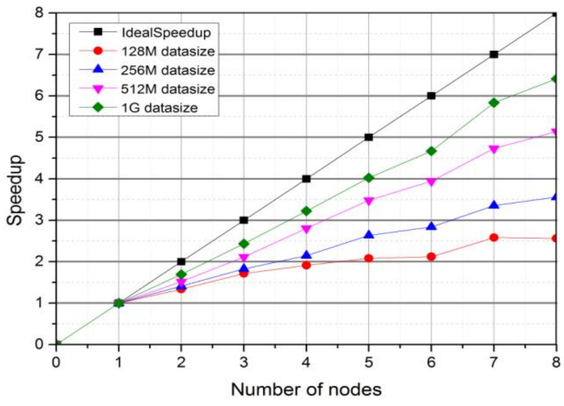


Fig. 8 Speedup of KASR.

Overall, these experimental results show that KASR performs well in accuracy, and KASR on Mapreduce framework has good scalability in “Big Data” environmrnt.

6 Related Work

There have been many recommender systems developed in both academia and industry. In [33], the authors propose a Bayesian-inference-based recommendation system for online social networks. They show that the proposed Bayesian-inference-based recommendation is better than the existing trust-based recommendations and is comparable to Collaborative Filtering recommendation. In [13], Adomavicius and Tuzhilin give an overview of the field of recommender systems and describe the current generation of recommendation methods. They also describe various limitations of current service recommendation methods, and discuss possible extensions that can improve recommendation capabilities and make recommender systems applicable to an even broader range of applications. Most existing service recommender systems are only based on a single numerical rating to represent a service's utility as a whole [34]. In fact, evaluating a service through multiple criteria and taking into account of user feedback can help to make more effective recommendations for the users.

With the development of cloud computing software tools such as Apache Hadoop, Map-Reduce, and Mahout, it becomes possible to design and implement scalable recommender systems in “Big Data” environment. The authors of [35] implement a CF algorithm on Hadoop. They solve the scalability problem by dividing dataset. But their method doesn't have favorable scalability and efficiency if the amount of data grows. [36] presents a parallel user profiling approach based on folksonomy information and implements a scalable recommender system by using Map-Reduce and Cascading techniques. Jin et al. [37] propose a large-scale video recommendation system based on an item-based CF algorithm. They implement their proposed approach in Qizmt, which is a .Net Map-Reduce framework, thus their system can work for large-scale video sites.

Generally speaking, comparing with existing methods, KASR utilizes reviews of previous users to get both of user preferences and the quality of multiple criteria of candidate services, which makes recommendations more accurate. Moreover, KASR on MapReduce has favorable scalability and efficiency.

7 Conclusions and Future Work

In this paper, we have proposed a keyword-aware service recommendation method, named KASR. In KASR, keywords are used to indicate users' preferences, and a user-based Collaborative Filtering algorithm is adopted to generate appropriate recommendations. More specifically, a keyword-candidate list and domain thesaurus are provided to help obtain users' preferences. The active user gives his/her preferences by selecting the keywords from the keyword-candidate list, and the preferences of the previous users can be extracted from their reviews for services according to the keyword-candidate list and domain thesaurus. Our method aims at presenting a personalized service recommendation list and recommending the most appropriate service(s) to the users. Moreover, to improve the scalability and efficiency of KASR in “Big Data” environment, we have implemented it on a MapReduce framework in Hadoop platform. Finally, the experimental results demonstrate that KASR significantly improves the accuracy and scalability of service recommender systems over existing approaches.

In our future work, we will do further research in how to deal with the case where term appears in different categories of a domain thesaurus from context and how to distinguish the positive and negative preferences of the users from their reviews to make the predictions more accurate.

Acknowledgement

W. Dou and J. Chen are the corresponding co-authors of this paper. This paper is partly supported by project National Science Foundation of China under Grant 91318301, and 61321491; National Key Technology R&D Program of the Ministry of Science and Technology under Grant 2011BAK21B06.

References

[1] J. Manyika, M. Chui, B. Brown, et al, “Big Data: The next frontier for innovation, competition, and productivity,” 2011.

[2] C. Lynch, “Big Data: How do your data grow?” Nature, Vol. 455, No. 7209, pp. 28-29, 2008.

[3] F. Chang, J. Dean, S. Ghemawat, and W. C. Hsieh, “Bigtable: A distributed storage system for structured data,” ACM Transactions on Computer Systems,Vol. 26, No. 2 (4) , 2008.

[4] W. Dou, X. Zhang, J. Liu, J. Chen, “HireSome-II: Towards Privacy-Aware Cross-Cloud Service Composition for Big Data Applications,” IEEE Transactions on Parallel and Distributed Systems, 2013.

[5] G. Linden, B. Smith, and J. York, “Amazon.com Recommendations: Item-to-Item Collaborative Filtering,” IEEE Internet Computing, Vol. 7, No.1, pp. 76-80, 2003.

[6] M. Bjelica, “Towards TV Recommender System Experiments with User Modeling,” IEEE Transactions on Consumer Electronics, Vol. 56, No.3, pp. 1763-1769, 2010.

[7] M. Alduan, F. Alvarez, J. Menendez, and O. Baez, “Recommender System for Sport Videos Based on User Audiovisual Consumption,” IEEE Transactions on Multimedia, Vol. 14, No.6, pp. 1546-1557, 2013.

[8] Y. Chen, A. Cheng and W. Hsu, “Travel Recommendation by Mining People Attributes and Travel Group Types From Community-Contributed Photos”. IEEE Transactions on Multimedia, Vol. 25, No.6, pp. 1283-1295, 2012.

[9] Z. Zheng, X Wu, Y Zhang, M Lyu, and J Wang, “QoS Ranking Prediction for Cloud Services,” IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No. 6, pp. 1213-1222, 2013.

[10] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, “Recommending and Evaluating Choices in a Virtual Community of Use,” In CHI '95 Proceedings of the SIGCHI Conference on Human Factors in Computing System, pp. 194-201, 1995.

[11] P. Resnick, N. Iakovou, M. Sushak, P. Bergstrom, and J. Riedl, “GroupLens: An Open Architecture for Collaborative Filtering of Netnews,” In CSCW '94 Proceedings of the 1994 ACM conference on Computer supported cooperative work, pp. 175-186, 1994.

[12] R. Burke, “Hybrid Recommender Systems: Survey and Experiments,” User Modeling and User-Adapted Interaction, Vol. 12, No.4, pp. 331-370, 2002.

[13] G. Adomavicius, and A. Tuzhilin, “Toward the Next Generation of Recommender Systems: A Survey of the Stateof- the-Art and Possible Extensions,” IEEE Transactions on Knowledge and Data Engineering, Vol.17, No.6 pp. 734-749, 2005.

[14] D. Agrawal, S. Das, A. El Abbadi, “Big Data and cloud computing: new wine or just new bottles?” Proceedings of the VLDB Endowment, Vol. 3, No.1, pp. 1647-1648, 2010.

[15] J. Dean, and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” Communications of the ACM, Vol. 51, No.1, pp. 107-113, 2005.

[16] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, “Dynamo: Amazons highly available key-value store,” In: Proceedings of the 21st ACM Symposium on Operating Systems Principles, pp. 205-220, 2007.

[17] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, “Dryad:Distributed data-parallel programs from sequential building blocks,” European Conference on Computer Systems, pp. 59-72, 2007.

[18] S. Ghemawat, H. Gobioff, and S. T. Leung, “The Google File System,” The 19th ACM Symposium on Operating Systems Principles, pp. 29-43, 2003.

[19] L. Zhang, “Editorial: Big Services Era: Global Trends of Cloud Computing and Big Data”. IEEE Transactions on Services Computing, Vol. 5, No. 4, pp. 467-468, 2012.

[20] Z. Luo, Y. Li and J. Yin, “Location: a feature for service selection in the era of big data,” 2013 IEEE 20th International Conference on Web Service, pp. 515-522, 2013.

[21] H. Schütze and J. O. Pedersen, “A cooccurrence-based thesaurus and two applications to information retrieval,” Information Processing & Management, Vol. 33, No. 3, pp. 307-318, 1997.

[22] Y. Jing and W. Croft, “An association thesaurus for information retrieval,” Proceedings of RIAO, Vol. 94, No. 1994, pp.146-160, 1994.

[23] B. Issac and W. J. Jap, “Implementing spam detection using bayesian and porter stemmer keyword stripping approaches,” TENCON 2009-2009 IEEE Region 10 Conference, pp. 1-5, 2009.

[24] P. Castells, M. Fernandez, and D. Vallet, “An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval,” IEEE Transactions on Knowledge and Data Engineering, Vol. 19, No.2, pp. 261-272, February 2007.

[25] Y. Zhu, Y. Hu,”Enhancing search performance on Gnutella-like P2P systems,” IEEE Transactions on Parallel and Distributed Systems, Vol. 17, No. 12, pp. 1482-1495, 2006.

[26] A. Chu, R. Kalaba, and K. Spingarn, “A comparison of two methods for determining the weights of belonging to fuzzy sets”, Journal of Optimization Theory and Applications, Vol. 27, No.4, pp.531-538, 1979.

[27] G. Salton, “Automatic Text Processing,” Addison-Wesley, 1989.

[28] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, “Item-based collaborative filtering recommendation algorithms,” Proceedings of the 10th international conference on World Wide Web, pp. 285-295, 2001.

[29] K. Lakiotaki, N. F. Matsatsinis, and A. Tsoukis,“Multi-Criteria User Modeling in Recommender Systems”, IEEE Intelligent Systems, Vol. 26, No. 2, pp. 64-76, 2011.

[30] Y. Pan, L. Lee, “Performance analysis for lattice-based speech indexing approaches using words and subword units,” IEEE Transactions on Audio, Speech, and Language Processing, Vol. 18, No. 6, pp. 1562-1574, 2010.

[31] G. Kang, J. Liu, M. Tang, X. Liu and B. cao, “AWSR: Active Web Service Recommendation Based on Usage History,” 2012 IEEE 19th International Conference on Web Services (ICWS), pp. 186-193, 2012.

[32] G. M. Amdahl, “Validity of the single-processor approach to achieving large scale computing capabilities,” Proceedings of spring joint computer conference, pp. 483-485, 1967.

[33] X. Yang, Y. Guo, Y. Liu, “Bayesian-inference based recommendation in online social networks,” IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No. 4, pp. 642-651, 2013.

[34] G. Adomavicius, and Y. Kwon, “New Recommendation Techniques for Multicriteria Rating Systems,” IEEE Intelligent Systems, Vol. 22, No. 3, pp. 48-55, 2007.

[35] Z. D. Zhao, and M. S. Shang, "User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop," In the third International Workshop on Knowledge Discovery and Data Mining, pp. 478-481, 2010.

[36] H. Liang, J. Hogan, and Y. Xu, “ Parallel User Profiling Based on Folksonomy for Large Scaled Recommender Systems: An Implimentation of Cascading MapReduce,” In Proceedings of the IEEE International Conference on Data Mining Workshops, pp. 156-161, 2010.

[37] Y. Jin, M. Hu, H. Singh, D. Rule, M. Berlyant, and Z. Xie, “MySpace Video Recommendation with Map-Reduce on Qizmt,” Proceedings of the 2010 IEEE Fourth International Conference on Semantic Computing, pp.126-133, 2010.

Shunmei Meng is currently working towards the PhD degree at the Department of computer Science and Technology, Nanjing University, China. She has received her Bachelor's degree in Computer Science from Nanjing University of Science and Technology. Her research interests include cloud computing, service computing, service recommendation, Big Data and MapReduce.

Wanchun Dou received his PhD degree in Mechanical and Electronic Engineering from Nanjing University of Science and Technology, China, in 2001. From Apr. 2001 to Dec. 2002, he did his postdoctoral research in the Department of Computer Science and Technology, Nanjing University, China. Now, he is a full professor of the State Key Laboratory for Novel Software Technology, Nanjing University, China. From Apr. 2005 to Jun. 2005 and from Nov. 2008 to Feb. 2009, he respectively visited the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, as a visiting scholar. Up to now, he has chaired three NSFC projects and published more than 60 research papers in international journals and international conferences. His research interests include workflow, cloud computing and service computing.

Xuyun Zhang is currently working towards the PhD degree at the Faculty of Engineering & IT, University of Technology, Sydney, Australia. Before joining UTS, he has received his Master's and Bachelor's degree in Computer Science from Nanjing University, China. His research interests include cloud computing, privacy and security, Big Data, MapReduce and OpenStack. He has published several papers in refereed international journals including IEEE Transactions on Parallel and Distributed Systems (TPDS).

Jinjun Chen is an Associate Professor from Faculty of Engineering and IT, University of Technology Sydney (UTS), Australia. He is the Director of Lab of Cloud Computing and Distributed Systems at UTS. He holds a PhD in Computer Science and Software Engineering from Swinburne University of Technology, Australia. Dr Chen's research interests include cloud computing, big data, workflow management, privacy and security, and related various research topics. His research results have been published in more than 100 papers in high quality journals and at conferences, including IEEE Transactions on Service Computing, ACM Transactions on Autonomous and Adaptive Systems, ACM Transactions on Software Engineering and Methodology (TOSEM), IEEE Transactions on Software Engineering (TSE), and IEEE Transactions on Parallel and Distributed Systems (TPDS). He received Swinburne Vice-Chancellor's Research Award for early career researchers (2008), IEEE Computer Society Outstanding Leadership Award (2008-2009) and (2010-2011), IEEE Computer Society Service Award (2007), Swinburne Faculty of ICT Research Thesis Excellence Award (2007). He is an Associate Editor for IEEE Transactions on Parallel and Distributed Systems. He is the Vice Chair of IEEE Computer Society's Technical Committee on Scalable Computing (TCSC), Vice Chair of Steering Committee of Australasian Symposium on Parallel and Distributed Computing, Founder and Coordinator of IEEE TCSC Technical Area on Workflow Management in Scalable Computing Environments, Founder and steering committee co-chair of International Conference on Cloud and Green Computing, and International Conference on Big Data and Distributed Systems.