IPHV7I20011X
International Journal Of Advanced Research and Innovation -Vol.7, Issue .II
ISSN Online: 2319 – 9253
Print: 2319 – 9245

A GENERALIZED METHOD OFQUALITY OF SERVICES BASED ALIGNMENT FORECASTING OVER CLOUD

[#1]V.Swathi-M.Tech Student,
[#2] R.Samaiah- Asst.Professor,

Department of Computer Science and Engineering,

Dr.K.V.Subba Reddy College Of Engineering For Women , Kurnool,A.P

Abstract: *Cloud computing is an internet based computing and is becoming popular. QoS rankings provide valuable information for making optimal cloud service selection from a set of functionally equivalent service candidates. To obtain QoS values, real-world invocations on the service candidates are usually required. To avoid the time-consuming and expensive real-world service invocation QoS ranking prediction framework is used. This framework requires no additional invocations of cloud services when making QoS ranking prediction. Two personalized QoS ranking prediction approaches are used to predict the QoS rankings directly. Comprehensive experiments are conducted employing real-world QoS data, including 300 distributed users and 500 real-world web services all over the world. The proposed uses modernized ranking approach which uses different QoS parameters to predict the ranking more accurately. Different QoS parameters like latency, availability, failure probability can be used to improve the ranking.* Building high Quality cloud applications becomes an immediately required research problem in cloud computing technology. Non-functional performance of cloud services is generally described by Quality-of-Service (QoS). To acquire QoS values, real-world usage of services candidates are generally required. At this time, there is no framework that can allow users to estimate cloud services and rank them based on their QoS values. This paper intends to framework and a mechanism that measures the quality and ranks cloud services for the users.

Keyword: *Quality-of-service, cloud service, ranking prediction, personalization.*

I.INTRODUCTION

Cloud computing is becoming popular. Building high-quality cloud applications is a critical research problem. QoS rankings provide valuable information for making optimal cloud service selection from a set of functionally equivalent service candidates. To obtain QoS values, real-world invocations on the service candidates are usually required. To avoid the time-consuming and expensive real-world service invocations, this paper proposes a QoS ranking prediction framework for cloud services by taking advantage of the past service usage experiences of other consumers. Our proposed framework requires no additional invocations of cloud services when making QoS ranking prediction. Two personalized QoS ranking prediction approaches are proposed to predict the QoS rankings directly. Comprehensive experiments are conducted employing real-world QoS data, including 300 distributed users and 500 real-world web services all over the world. The experimental results show that our approaches outperform other competing approaches.

Cloud computing is becoming popular. Building high-quality cloud applications is a critical research problem. QoS rankings provide valuable information for making optimal cloud service selection from a set of functionally equivalent service candidates. To obtain QoS values, real-world invocations on the service candidates are usually required. To avoid the time-consuming and expensive real-world service invocations, this paper proposes a QoS ranking prediction framework for cloud services by taking advantage of the past service usage experiences of other consumers. Our proposed framework requires no additional invocations of cloud services when making QoS ranking prediction. Two personalized QoS ranking prediction approaches are proposed to predict the QoS rankings directly. Comprehensive experiments are conducted employing real-world QoS data, including 300 distributed users and 500 real-world web services all over the world. The experimental results show that our approaches outperform other competing approaches.

Recent days the cloud computing technology is popular because it is an attracting technology in the field of computer science. Cloud computing is internet based computing that generally referred the shared configurable resources (e.g., infrastructure, platform, and software) is provided with computers and other devices as services. Cloud computing entrusts services with a customer's data, software and computation over a network. The customer of the cloud can get the services through the network. In other words, users are using or buying computing services from others. Cloud can provide Anything as a Service (AaaS). In Cloud technology the QoS based service selection is an essential research topic. When many services offer similar functionality QoS values show a critical role for separating the optimal service for that particular task [8]. Because many number of cloud services are available. Since the user points of view, it is difficult to choose the best service and what mechanism used to select their services [6]. Qos models are associated with End-Users and providers.

In existing system Component-based system [15], cloud applications usually involve several cloud components communicating with each other over application programming interfaces, such as through web services. The process of this cloud application is collected by a number of software components, where each component fulfils a specified functionality. While there are a number of functionally equivalent services in the cloud, optimal service selection becomes essential. Once construct the best cloud service selection from a set of functionally the same services, QoS values of cloud services give key information to help decision making. Software components are invoked locally, whereas in cloud applications.

Cloud services are invoked remotely by Internet connection. Client-side performance of cloud services is thus seriously influenced by the unpredictable Internet connections. Therefore, different cloud applications may receive dissimilar levels of quality for the matching cloud service.so it need the additional invocations of cloud services**.** But it has following cons:

(1) When the number of applicant services is huge, it is complicated for the cloud application designer to estimate all the cloud services resourcefully

(2) QoS is very low so Improve the overall quality, by replacing the low quality components with better ones.

(3) It does not guarantee that the employed services will be ranked correctly.

Our proposed paper overcome above problems using Personalized ranking prediction framework, named Cloud Rank, it is the first personalized ranking prediction framework to calculate the QoS ranking of a set of cloud services not including requiring in addition real-world service invocations from the intended users. This approach takes gain of the past usage experiences of other users for building personalized ranking prediction for the Active user. It use the two algorithm namely cloudrank1.

This paper overcomes the existing system and it has the following pros:

(1) It avoids time-consuming and expensive real-world service invocations. (2) It does not require additional invocations of cloud services.

(3) It takes the advantage of past usage experiences from other users.

(4) Identify the dangerous trouble of personalized QoS ranking for cloud services and proposes a QoS ranking prediction framework to tackle the problem.

(5) Achieve better ranking accuracy for cloud services compared with other ranking algorithm.

(6) Publicly release this service QoS data set for future research, so make this experiment reproducible.

II. RELATED WORK

Although the traditional UDDI standard 2 does not refer to QoS for web services,many proposals have been devised to extend the original model and describe webservices' quality capabilities, e.g., QML, WSLA and WSOL [10]. The issue oftrust and reputation management in Internet-based applications has also beena well-studied problem [1, 2]. The UX architecture [11] suggests using dedicated servers to collect feedbackof consumers and then predict the future performance of published services.

2 Related Work Although the traditional UDDI standard 2 does not refer to QoS for web services, many proposals have been devised to extend the original model and describe web services' quality capabilities, e.g., QML, WSLA and WSOL [10]. The issue of trust and reputation management in Internet-based applications has also been a well-studied problem [1, 2]. The UX architecture [11] suggests using dedicated servers to collect feedback of consumers and then predict the future performance of published services. Our QoS provisioning model is grounded on previous work of [3, 4, 11, 13,14]. The trust and reputation management of our work is most similar to thatof [17, 18] but we employ the idea of distrust propagation more accurately byvobsering vthat trust information from a user report can also be used to reveal dishonesty of other reporters and by allowing this distrust to be propagated to similar ones. Other ideas of the trust and reputation management method are based on [19–21]. However, these reputation management techniques are still simple and not robust against various cheating behaviors, e.g., collusion among providers and reporters with varying actions over time. Consequently, the quality of the searching results of those discovery systems will not be assured if there are lots of colluding, dishonest users trying to boost the quality of their own services and badmouth about the other ones. [16] suggests augmenting service clients with QoS monitoring, analysis, and selection capabilities, which is a bit unrealistic as each service consumer would have to take the heavy processing role of a discovery and reputation system. Other solutions [3{7] use third-party service brokers or specialized monitoring agents to collect perfor- mance of all available services in registries, which would be expensive in reality. Though [7] also raises the issue of accountability of Web Service Agent Proxies in their system, the evaluation of trust and reputation for these agents is still an open problem.

There have been many studies of Quality-of-Service for cloud services. Since this work explores the issue of building high quality cloud applicattions.Quality-of-Service (QoS) is usually describing the non-functional characteristics of services and employed as an important differentiating point of different Web services. Users in different geographic locations collaborative with each other to evaluate the target Web services and share their observed Web service QoS information. Areas related to this work include the following: QoS Evaluation of Web Services, Neighborhood-based QoS Prediction of Web Services, and Model-based QoS Prediction of Web Services.

2.1 QoS Evaluation of Web Services

To accomplish efficient Web service evaluation, we recommend a distributed QoS evaluation framework for Web services. This framework employs the idea of user- collaboration, which is the means the concept of Web 2.0. In our framework, users in different geographic locations distribute their observed Web service QoS information. That information is stored in a centralized server and will be reuse for any other users.

2.2 Neighborhood-based QoS Prediction of Web Services

To exactly predict the Web service QoS values, we suggest a neighborhood-based collaborative filtering approach for predict the QoS values for the active user by employ past Web service QoS data from other similar users. Our approach systematically combine the user based approach and the item-based approach and it requires no Web service invocations and can help service users find out appropriate Web services by analyze QoS information from their similar users.

2.3 Model-based QoS Prediction of Web Services

The neighborhood-based QoS prediction approach has several drawbacks, including (1) the computation complexity is too high, and (2) it is not easy to find similar users/items when the user-item matrix is very sparse. To address these drawbacks, we plan a neighborhood-integrated matrix factorization (NIMF) approach for Web service QoS value prediction. Our approach explores the social wisdom of service users by systematically fusing the neighborhood based and the model-based collaborative filtering approaches to achieve higher prediction accuracy. Item-Based Top-N Recommendation Algorithms that determine the similarity among the different items from the set of items to be suggested. The steps in this class of algorithms are (i) the method used to calculate the similarity between the items, and (ii) the method used to combine these similarities in order to calculate the similarity between a bin of items and a candidate recommender item. The goal of top-N recommendation algorithm was to categorize the items purchase by an individual consumer into two classes: like and dislike. This algorithm is faster than the conventional user-neighborhood based recommender systems and it provide recommendation with comparable or better quality. The proposed algorithms are independent of the size of the user–item matrix [1]. Automatic Weighting Scheme for Collaborative Filtering that automatically computes the weights for different items based on their ratings from training users. The new weighting scheme will create a clustered distribution for user vectors in the item space by bringing users of similar interest's closer and separating users of different interests more distant but it provides low performance than Pearson Correlation Coefficient method [2].

The Collaborative Filtering technique that predict the missing data. It is making automatic predictions (filtering) about the interests of a user by collecting taste information from many other users (collaborating). User-based collaborative filtering predicts the ratings of active users based on the ratings of similar users found in the user-item matrix, Item-based collaborative filtering predicts the ratings of active users based on the information of similar items computed but it increases the density of User-Item Matrix and it predict some of the missing data only [18]. Collaborative filtering approach that addresses the item ranking problem directly by modeling user preferences derived from the ratings. It performs ranking items based on the preferences of similar users and it is used to identifying and aggregating the preferences in order to produce a ranking of items but it need to including data smoothing for improving traditional rating oriented collaborative filtering and then it has to utilize content information to our ranking-oriented approach [19].

III. Qos RANKING PREDICTION

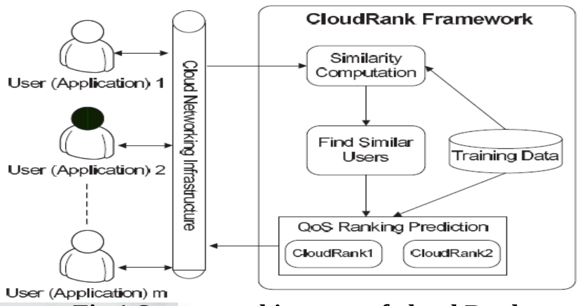

Fig.1 System architecture of cloud Rank

This section presents our Cloud Rank QoS ranking prediction framework for cloud services. Section 3.1 calculates the similarity of the active user with training users based on their rankings on the commonly invoked cloud services identifies a set of similar users presents two QoS ranking prediction algorithms, named Cloud Rank1 and Cloud Rank2, respectively analyses the computational complexity.

3.1Similarity Computation

Ranking similarity computations compare users QoS rankings on the commonly invoked services. Suppose we have a set of three cloud services, on which two users have observed response-times (seconds) of {1, 2, 4} and {2, 4, 5}, respectively. The response-time values on these services observed by the two users are clearly different nevertheless; their rankings are very close as the services are ordered in the same way. Given two rankings on the same set of services, the Kendall Rank Correlation Coefficient (KRCC) evaluates the degree of similarity by considering the number of inversions of service pairs which would be needed to transform one rank order into the other. The KRCC value of user's u and v can be calculated by

A. Similarity Computation

The similarity computation of active users and training users are calculated based on the user provided qos values using Kendall Rank Correlation Coefficient (KRCC).It evaluates the degree of similarity by considering the number of inversions of service pairs which would be needed to transform one rank order into the other. The KRCC value of user's u and v can be calculated by,

$$\mathrm{Sim}(u,v) = \frac{C-D}{\frac{N(N-1)}{2}} \qquad (1)$$

Where N is the number of services, C is the number of concordant between two lists, D is the number of discordant pairs, and there is totally N (N-1) /2 pairs for N cloud services. Ranking similarity is determined between the users. The response-time values on set of cloud services observed by the users are different.

B. Find Similar Users

Set of similar users can be identified to the Active user. Information of all the users for making ranking prediction, which may include dissimilar users. QoS values of dissimilar users will greatly influence the prediction accuracy. In our approach, a set of similar users is identified for the active user u by,

$$N(u) = \{v | v \epsilon T \ , Sim(u, v) > 0, \quad \neq u\} \quad (2)$$

Where T_u is a set of the Top-K similar users to the user u and Sim (u, v) >0 excludes the dissimilar users with negative similarity values. The value of Sim (u, v) in 2 is calculated by (1)

C. Personalized Service Ranking

First predict the missing QoS values before making QoS ranking. Accurate QoS value is predicted using rating-oriented collaborative filtering approach. It does not lead to accurate QoS ranking prediction use two ranking algorithm.

D. Provide the Service to Active User

Personalized service ranking takes the advantage of past usage experiences of similar users. Then ranking prediction results are provided to the active user. Further exact ranking prediction results can be achieved through providing QoS values on more cloud services.

3.2 Find Similar Users

By calculating similarity values between the current active user with other training users, the similar users can be identified. Previous approaches usually employ information of all the users for making ranking prediction of the current user, which may include dissimilar users. However, employing QoS values of dissimilar users will greatly influence the prediction accuracy.

3.3 QoS Ranking Prediction

Rating-oriented collaborative filtering approaches first predict the missing QoS values before making QoS ranking. The target of rating-oriented approaches is to predict QoS values as accurate as possible. However, accurate QoS value prediction may not lead to accurate QoS ranking prediction. For example, assume the expected response times of three services are 2, 3, and 5 seconds, respectively. There are two predictions using rating-oriented approaches: (3, 2, 4) and (1, 2, 3). Since rating-oriented approaches try to predict the QoS value as accurate as possible, Prediction 1 is better than Prediction 2, since it has a smaller MAE value. However, from the ranking-oriented perspective, Prediction1 is worse than Prediction 2 since the former leads to incorrect ranking based on the predicted QoS values. To address this problem, we propose two ranking- oriented approaches, named as Cloud Rank1 and CloudRank2, in the following. Our ranking-oriented approaches predict the QoS ranking directly without predicting the corresponding QoS values.

IV. ALGORITHM

QoS-based Service Selection and Ranking

Our QoS-enabled distributed service discovery framework is presented in de- tail in [8]. Quality properties of web services are described by concepts from a QoS ontology and then embedded into service description ¯les using techniques suggested by WS-QoS [3] and Ran [4]. The value of a quality parameter of a web service is supposed to be *normalized* to a non-negative real-valued number regarding service-speci¯c and call-speci¯c context information where *higher nor- malized values represent higher levels of service performance*. We are aware that the issue of normalizing the values of various quality attributes is complicated, but this is out of the scope of our current study. However, with most frequently used QoS concepts, e.g., reliability, execution-time, response-time, availability, etc., the answer is well-de¯ned and straight-forward. For instance, a web service with a normalized QoS parameter value for reliability of 0:90 will be considered as more reliable to another one with a normalized reliability value of 0:50. In this case the normalized reliability is measured as its degree of being capable of maintaining the service and service quality over a time period *T*. The ontology language we use to describe service semantics and to de¯ne the QoS ontology is WSMO 3, but other models, e.g., OWL-S 4, would also be applicable. For ex- perimental evaluations, we have developed a simple QoS ontology for the VISP use case including the most relevant quality parameters for many applications, i.e., availability, reliability, execution time, etc. We currently assume that users and providers share a common ontology to describe various QoS concepts. How- ever, this could be relaxed with the help of many existing ontology mapping frameworks.

Predicting Service Performance

In order to predict the quality of a web service *Si*, we collect all QoS feed- backs on its performance over a time period *W* and use a *real-valued time series forecasting technique* to predict its future quality conformance from past data. To understand the concepts in our algorithms we start with two fundamental denitions. In order to ¯lter out as much dishonest reports as possible and to take only the most credible ones in the QoS predicting process, we apply our trust and rep utation management techniques comprising of two steps: a report preprocessing and a report clustering phase. The ¯rst phase evaluates the credibility of collected user reports by applying a trust-and-distrust propagation approach, which relies on some initial trusted reports produced by special monitoring agents. We consider two QoS reports as *comparable* if they are related to the same service during a speci¯c time interval ±*t* and as *incomparable* otherwise. Generally, we can set this ±*t* as big as the length of the period during which the correspond- ing service provider does not change the promised quality values of this service. Two *comparable* QoS reports are considered to be *similar* if the squared Euclid- ean distance between their conformance vectors is less than a speci¯c threshold. On the contrary, they are regarded as *dissimilar* if this distance is greater than another threshold value.

The report preprocessing step is done according to Algorithm 1. *nch* and *nh*1*; nh*2 (*nh*1 < *nh*2) are threshold values to estimate a user as cheating or honest regarding to the similarity of its reports to other cheating/honest ones (line 9*;* 17 and 18). *N* and *T* represent for how long and how frequent a user stay in and submit QoS reports to the system. The values of *nh*1*; nh*2*; nch;N* and *T* are design parameters to be chosen depending on properties of the collected reports after running the algorithm several times. Generally, higher values of these parameters stand for higher level of caution when estimating behaviors of users regarding current evidences against them. After ¯nishing the preprocessing phase, we can identify a certain number of cheaters and honest users. However, this trust-distrust propagation phase may not be able to evaluate the credibility of all reports in case the user communities of certain services are isolated from other communities as well as if we set the values of *nh*1*; nh*2 and *nch* too high in Algorithm 1. Therefore, in the next step we have to estimate the trustworthiness of the remaining reports of which credibility has not been evaluated .

Algorithm

```
Qos Reports Preprocessing()
    1.  all trusted agents are marked as honest users;
    2.  all reports of trusted agents are marked honest;
         repeat
    3.  all unmarked reports of each cheating user are
        marked cheating;
         for each unmarked report do
    4.  if this report is dissimilar from an honest report then
        mark it cheating;
    5.  if this report is similar with a cheating report then
        mark it cheating;
    6.  end for
    7.  users with at least nch reports similar with cheating
        ones are marked cheating;
    8.  users with at least N reports in at least T di®erent
        time are marked stable;
    9.  until there is no new cheating user discovered;
    10. repeat
    11. all unmarked reports of each honest user are marked
        as honest;
          for each unmarked report and each report marked
        cheating do
    12. if this report is similar with an honest report then
        mark it honest;
    13. end for
         unmarked users with at least nh1 reports similar
         with honest ones are marked honest;
    14. users marked as cheating and having at least nh2
        reports similar with honest
    15. ones are re-marked honest;
    16. until there is no new honest user discovered;
```

After the trust and reputation evaluation in the two above phases, for each service *Si*, we will have a list *Gi* of groups of reports on QoS of *Si* over time. Generally, *Gi* includes the groups containing those reports that were previously marked as honest/cheating during the trust-distrust propagation phase, as well
as other clusters of reports obtained after the clustering step. We will assign the credibility *wg*
i of a report group *gi 2 Gi* as follows. Firstly, we ¯lter out all dishonest ratings by assign *wg i* = 0*:*0 for all groups of reports marked as cheating during the trust-distrust propagation. If there exists the group *g*0
i of reports previously marked honest during that step, we assign *wg*0 *i* = 1*:*0 whereas letting
wg i = 0*:*0 for the remaining groups. Otherwise, we try to compute *wg i* so that this value would be proportional to the probability that the group *gi* is trustworthy among all of the others. Our heuristic is to assign higher weight to clusters which are populated more densely, having bigger size and with more stable users. This assumption is reasonable, as the reports of independent cheaters are likely to be scattered, and in the case liars cooperate with each other to cheat the system, the size of their corresponding clusters will not exceed those consisting only of honest reports as it would be too costly to dominate the system with numerous and clustered dishonest ratings. Even if dishonest providers try to produce lots of more condense reports so that they could get high in°uences to the ¯nal ranking
results at any cost, these values will be separated from honestly reported values and therefore are likely to be discovered during the distrust-propagation phase (line 3 to line 11 in Algorithm 1), provided we have enough trustworthy reports to use. Speci¯cally, *wg*
i could be estimated based on the following information: the number of users in the cluster *gi* (*sizeg*
i), the number of all users producing reports in all clusters of *Gi* (*allusersi*), the number of stable users in this cluster (*stableg*
i), the total number of stable users in all clusters of *Gi* (*allstablei*), as well as the average distance *dg i* from the member reports of cluster *gi* to its centroid values. Based on our model in section 3, the credibility of one report would depend on the distance between its conformance vector and that of an honest report. Therefore, the similarity among credibility of di®erent reports
in one cluster *gi* would be inversely proportional to its *dg i* value. Furthermore, a report in *gi* would be honest in two cases: (1) it is submitted by a *stable and honest* user; (2) it is produced by an *unstable and honest* user. Let *Pstbl* and *Punstbl* be the probability that this report is of a stable user and of an unstable user, respectively, and let *Pstblcr* and *Punstblcr* be the probability that stable and unstable users report credibly, then we have

Punstblcr could be estimated by comparing reports of trusted agents with those of sample stable/unstable users to derive an appropriate value at the whole network level. The value of *C* represents our belief in the relation between the density of a report cluster and the credibility of its members, which is considered as parameters of the reputation system and to be set by experience. The future quality conformance ^ *Cij* of a service *Si* in providing a QoS at- tribute *qij* is predicted using a linear regression method, thus we have: ^ *Cij* = *LinearRegression*(*Ct*
ij), *t 2 f*0*; ±t; : : : ;* (*W ¡* 1)*:±tg*, where *C*
t
ij is the evaluated
QoS conformance value of the quality parameter *qij* at time *t*. We compute *C* as the average of conformance values reported by various user groups in the sys- tem at that speci¯c time point, using the evaluated credibility *wg i* s as weights in where *Ct* is the mean of conformances of a report group *gi* on *Si*'s quality attribute *qij* at time *t*, i.e., a centroid value of a cluster/group of reports produced after the trust and reputa- tion evaluation phase. Regarding the probabilistic behavior of users and services as in section 3, we consider ^ *Cij* as an approximate estimate of the expecte.

QoS-based Service Selection and Ranking

We define QoS requirements in a user query as a vector *Q* of triples *fqj ; nj ; vjg*where *qj* represents for the required QoS attribute, *nj* is the level of importance of this quality attribute to the user and *vj* is the minimal delivered QoS value that this user requires. To rank services according to its prospective level of satisfying user's QoS requirements, we utilize the Simple Additive Weighting method, which produces ranking results very close to those of more sophisticated Decision Making techniques [5]. Thus, the QoS rank of a service *Si* in fullling.

```
Algorithm 2 Qos Selection Ranking(ServiceList L, ServiceQuery Q)

1: Derive the list of QoS requirements in Q: Lq = f[q1; n1; v1]; :::; [qs; ns; vs]g
2: Initialize Score[Sij ] = 0:0 for all services Sij 2 L;
3: for each quality concept qj 2 Lq do
4: for each service Sij 2 L do
5: Search the list Lqos of qj for Sij ;
6: if Sij is found then
7: Score[Sij ] = Score[Sij ] + nPj :wij

8: else
9: Remove Sij from L;
10: end if
11: end for
12: end for
13: Return the list L sorted in descending order by Score[Sij ] s;
```

V. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a new QoS-based web service selection and ranking algorithm with trust and reputation management support. We have shown that our selection and ranking solution yields very good results in most cases. As the proposed reputation management mechanism is robust against various cheating behaviors, the results are generally of good quality even in hostile situations in which many di®erent types of cheaters make up a high percentage of the overall users and report values with remarkable variances. Personalized QoS ranking prediction framework for cloud services, which requires no additional service invocations when making QoS ranking. By taking advantage of the past usage experiences of other users, our ranking approach identifies and aggregates the preferences between pairs of services to produce a ranking of services. We propose two ranking prediction algorithms for computing the service ranking based on the cloud application designer's preferences. Experimental results show that our approaches outperform existing rating-based approaches and the traditional greedy method.

For future work, like to improve the ranking accuracy of our approaches by exploiting additional techniques (e.g., data smoothing, random walk, matrix factorization, utilizing content information, etc.). When a user has multiple invocations of a cloud service at different time, we will explore time-aware QoS ranking prediction approaches for cloud services by employing information of service users, cloud services, and time. As our current approaches only rank different QoS properties independently, we will conduct more investigations on the correlations and combinations of different QoS properties. We will also investigate the combination of rating-based approaches and ranking-based approaches, so that the users can obtain QoS ranking prediction as well as detailed QoS value prediction. Moreover, we will study how to detect and exclude malicious QoS values provided by users.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," Technical Report EECS-2009-28, Univ. California, Berkeley, 2009.
[2] K.J. arvelin and J. Kekalainen, "Cumulated Gain-Based Evaluation of IR Techniques," ACM Trans. Information Systems, vol. 20, no. 4, pp. 422-446, 2002.
[3] P.A. Bonatti and P. Festa, "On Optimal Service Selection," Proc. 14th Int'l Conf. World Wide Web (WWW '05), pp. 530-538, 2005.
[4] J.S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," Proc. 14th Ann. Conf. Uncertainty in Artificial Intelligence (UAI '98), pp. 43-52, 1998.
[5] R. Burke, "Hybrid Recommender Systems: Survey and Experi- ments," User Modeling and User-Adapted Interaction, vol. 12, no. 4, pp. 331-370, 2002.
[6] W.W. Cohen, R.E. Schapire, and Y. Singer, "Learning to order things," J. Artificial Intelligent Research, vol. 10, no. 1, pp. 243-270, 1999.
[7] M. Deshpande and G. Karypis, "Item-Based Top-n Recommenda- tion," ACM Trans. Information System, vol. 22, no. 1, pp. 143-177, 2004.

AUTHORS PROFILE:

[1]. V.Swathi studying M.Tech, received b.tech degree in Computer Science And Engineering from Dr.k.v.subba reddy college of engineering for women in 2012,Kurnool

[2]. R.Samaiah, M.Tech (MISTE) received his B.Tech Degree in Computer Science and Engineering from Sri Venkateshwara University , Thirupathi, india in the year 2005 and M.Tech in Computer Science from Vishweswaraiah Technological University , india in the year 2008, presently working as Asst.Professor at Dr.K.V.S.R.C.E.W,Kurnool, India. His research areas includes computer networks.