Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Department of Software

QoS Prediction for Web Service Recommendation Based on Reputation and Location Clustering-Aware Collaborative Filtering

A Dissertation
Submitted to the Council of the College of Information Technology for Postgraduate Studies of University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information Technology- Software

Muayad Najm Abdullah Yaseen

Supervised by

Prof. Dr. Wesam S. Bhaya

2021 A.D. **١٤٤٣ A.H.**

بسم الله الرحمن الرحيم

﴿ قَالُوا سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا ۖ إِنَّكَ أَنتَ الْعَلِيمُ الْحَكِيمُ (٣٢) ﴾

صدق الله العلي العظيم
سورة البقرة: الآية (٣٢)

Declaration

I hereby declare that this dissertation, submitted to University of Babylon in partial fulfillment of requirements for the degree of Doctorate of Philosophy in Information Technology-Software has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose sources are appropriately cited in the references.

Signature:

Name: Muayad Najm Abdullah

Date: / / 2021

Supervisor Certification

I certify that this dissertation titled "**QoS Prediction for Web Service Recommendation Based on Reputation and Location Clustering-Aware Collaborative Filtering**" was prepared under my supervision at the Department of Software / College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Doctor of Philosophy in Information Technology- Software.

Signature:
Name: Prof. Dr. Wesam S. Bhaya
Date: / / 2021

The Head of the Department Certification

In view of the available recommendations, I forward the dissertation entitled "**QoS Prediction for Web Service Recommendation Based on Reputation and Location Clustering-Aware Collaborative Filtering**" for debate by the examination committee.

Signature:
Name: Asst. Prof Dr. Ahmed Saleem Abbas
Date: / / 2021

Certification of the Examination Committee

We, the undersigned, certify that (Muayad Najm Abdullah) candidate for the degree of Doctor of Philosophy in Information Technology-Software, has presented his dissertation of the following title (**QoS Prediction for Web Service Recommendation Based on Reputation and Location Clustering-Aware Collaborative Filtering**) as it appears on the title page and front cover of the dissertation that the said dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: (19/10/2021).

Signature:
Name: Dr. Huda Naji Nawaf
Title: Professor
Date: / / 2021
(**Chairman**)

Signature:
Name: Dr. Soukaena Hassan Hashen
Title: Professor
Date: / / 2021
(**Member**)

Signature:
Name: Dr. Hawraa Hassan Abbas
Title: Professor
Date: / / 2021
(**Member**)

Signature:
Name: Dr. Mazin Sameer Ali
Title: Assistant Prof**essor**
Date: / / 2021
(**Member**)

Signature:
Name: Dr. Ali Hadi Hasan
Title: Assistant Professor
Date: / / 2021
(**Member**)

Signature:
Name: Dr. Wesam S. Bhaya
Title: Professor
Date: / / 2021
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Dr. Hussein Atiya Lafta
Title: Professor
Date: / / 2021
(**Dean of Collage of Information Technology**)

Dedication

I dedicate the accomplishment of this dissertation to the my family.

Muayad

Acknowledgments

First of all, I would like to thank ***Allah***, The Creator, for His help and acceptance of my prayers that make the accomplishment of this work more than a dream after all what happened.

I would like to express my deep thanks and gratitude to my supervisor Prof. Dr. Wesam S. Bhaya for his valuable guidance and encouragement during the development of my work.

I would also like to thank the staff members of Information Technology College for their faithful efforts to give us the most scientific topics and endless support in all directions aiming to bring into perfection their scientific followers, and for their unforgettable kind and wise management to our affairs during the research period. I'm very grateful and thankful to the Dean of Information Technology College Prof. Dr. Hussein Atiya Lafta.

Last but not least, words cannot express my gratitude and indebtedness to my family, especially my sympathetic, compassionate and beloved mother, my dear brothers, my sister, my dear wife, and my beloved sons. Words cannot describe their constant love, care, concern, patience, and direction in every aspect of my life throughout the years of my study. I'm forever thankful, grateful, and indebted to them.

Muayad Najm Abdullah

Abstract

In recent years, cloud computing has become more popular and scalable services. The core component of this technology is the web service. However, Quality of Service (QoS) for web services seems to have an essential role when it comes to select the best web services from a large number of web services with similar functionality. Therefore, evaluating the user-side efficiently in terms of quality of web services has become a critical research topic.

In this dissertation, a hybrid approach is proposed that combines a context-based method with a collaborative filtering technique to predict the QoS values of web services and select the best services for active users based on historical observations of other users' services. However, in the proposed approach, two important contextual information were used: the reputation and the geographic location of the users. The reputation is used to improve the accuracy of the prediction, while the geographical location is used to improve the similarity calculation of the users and to mitigate data sparsity problem.

The proposed parts of this work have been divided into three stages. In the first stage, a set of untrusted users is identified through the Dirichlet probability distribution on the basis of the user's reputation, followed by processing the unreliable data provided by dishonest users using the mean of the reliable interval in each service. The second stage, clustering all users by geographic location, of which, the users that located in the same cluster have similar QoS values for the same services. In the third stage, the unknown QoS values are predicted using a collaborative filtering approach based on the similarity between users in each cluster.

The dissertation results were very satisfactory compared to the previous collaborative filtering approaches for predict unknown QoS values. The experimental results show that the proposed approach significantly improves the performance of QoS prediction in terms of efficiency and accuracy. The measures of MAE and RMSE for the response time attribute were 0.3601 and 1.0809, respectively, and the measures of MAE and RMSE for the throughput attribute were 10.746 and 39.867, respectively.

Declaration Associated with this Thesis

Some of the works presented in this dissertation have been published as listed below.

1. The work entitled "Predication of Quality of Service (QoS) in Cloud Services: A Survey" has been published in International Conference of Modern Applications on Information and Communication Technology 2020 (ICMAICT), IOP Publishing. The classification of this publishing is related to the Scopus database.

2. The work entitled "Predicting QoS for Web Service Recommendations Based on Reputation and Location Clustering with Collaborative Filtering" has been published in 7th International Conference on Contemporary Information Technology and Mathematics 2021 (ICCITM), IEEE Publishing. The classification of this publishing is related to the Scopus database.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

Abbreviation	Meaning
AS	Autonomous System
CaaS	Computation as a Service
CF	Collaborative Filtering
Cloud-Pred	Cloud Prediction
DaaS	Data as a Service
DBSCAN	Density Based Spatial Clustering of Applications with Noise
IaaS	Infrastructure as a Service
IoS	Internet of Services
IP	Internet Protocol
IPCC	(Item-based collaborative filtering method using Pearson Correlation coefficient
IT	Information Technology
MAE	Mean Absolute Error
MF	Matrix Factorization
NIST	National Institute of Standards and Technology
NMF	Nonnegative Matrix Factorization

PaaS	Platform as a Service
PCC	Pearson Correlation Coefficient
QoS	Quality of Services
RMSE	Root Mean Squared Error
RS	Recommender System
SaaS	Software as a Service
SOA	Service-oriented architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery, and Integration
UIPCC	User-Item-based collaborative filtering method using Pearson Correlation Coefficient
UPCC	User-based collaborative filtering method using Pearson Correlation Coefficient
WS	Web Service
WSDL	Web Services Description Language

Chapter One

General Introduction

1.1 Introduction

Recently, many technologies such as Software Oriented Architecture (SOA) and Cloud Computing have found more popularity [1] [2]. The core component of these technologies is web service [3]. The services offered on the web expanded rapidly and founded the web services needed by the users, and recommender system have played a significant role [4]. However, the increasing growth of web services has resulted in a large number of web services with equivalent functionality [5]. Therefore, selecting the best web service from a huge number of web services with the same functionality is one of the challenges in SOA [6] .

Quality of service (QoS) is a quality measure that refers to a set of features for evaluating the performance of candidate services. However, QoS of web services plays an important role in selecting the best service [7]. Therefore, QoS becomes an important research topic in cloud computing for service selection [8] [9].

In a cloud environment, there are two different types of attributes which utilized to evaluate Quality of Services (QoS) for cloud computing services, functional (static) and non-functional (dynamic) attributes. Functional (static) attributes refer to the performance of QoS services on the service provider side such as accuracy, cost, and reliability that is usually unchangeable for different users, while non-functional (dynamic) attributes refer to QoS observation for invocation services at the user side such as response time, throughput, which is not identical for different users due to the unpredictable links at network environment and diverse in contexts [10] [11].

QoS attributes have been identified as the main non-functional features of web services and the need for their application when selecting

web services [12]. On the user-side, it is necessary to conduct a personalized Quality of Service evaluation for each user. However, the user will typically invoke a small number of services to obtain their QoS performance [13]. Since the evaluation of services on the user-side requires the use of all services on the user-side, which is practically impossible. Therefore, many approaches have been developed to accurately predict unknown QoS values of services using historical invocation records of different users.

Collaborative filtering (CF) technique has widely been used in commercial recommendation system for predicting user interests. It recommends services based on similarity measures between users and/or services. However, this technique make prediction based on past observation of similar users [14].

In some cases, users may be unfairly prejudiced when providing QoS values for invoking services. Since such users do not reflect the actual value of the QoS attributes. Therefore, in CF -based techniques, the unreliability of QoS data provided by untrusted users leads to inaccurate prediction, so checking the trustworthiness of users is an important issue to reduce the impact of unreliable data on prediction accuracy [15] [16].

This dissertation presents a robust model for building personalized QoS prediction by using past invocation records based on reputation and geographic information as context to improve prediction accuracy.

1.2 Motivation

Web services in cloud computing with equivalent functionality have two types of attributes: static (functional) and dynamic (non-functional). To display the QoS values of web services, they should be

distinguished individually according to their static or dynamic nature. The static attributes (functional) of each service, such as the cost of the services, are the same for all users, for example, if we perform a traditional search, we will observe the same QoS values. However, for all other dynamic attributes (non-functional), the value of each QoS attribute of web service can vary for each user, depending on the environment of the Internet and the location of the user. On the other hand, dynamic (non-functional) attributes are the QoS values observed by different users of services at the user-side. So, QoS values can be unreliable data provided by untrusted users for some reasons.

To address these problems, a personalized CF-based approach is proposed to predict QoS values in web services in light of the reputation and geographic information using the historical QoS values of other users who invoked the service.

1.3 Problem Statement

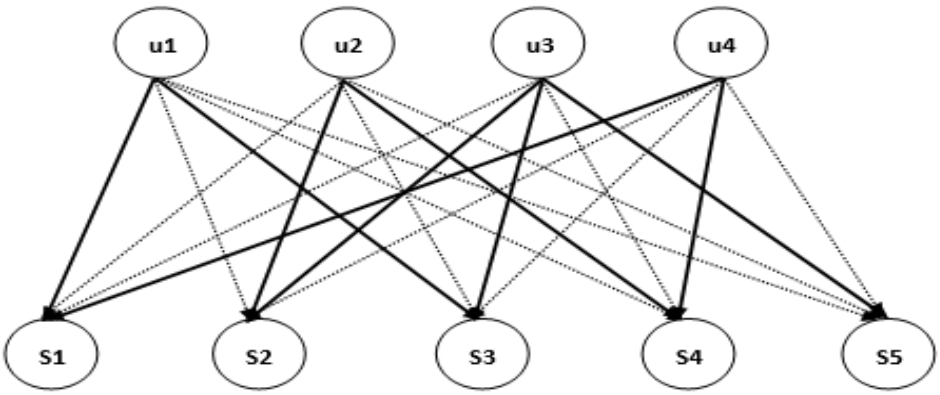
In this section, we will formally state the problem of predicting the QoS values of web services. The QoS values observed by different users of services at the user-side can be represented by a user-service matrix that gives an entry for each user-service pair that represents user's observed QoS value for that service, in which case the problem will be as follows:

1. $U = \{u_1, u_2, \dots, u_m\}$ is the set of users of the web services, where u_i $(1 \leq i \leq m)$ represent a service user.
2. $S = \{s_1, s_2, \dots, s_n\}$ is the set of web services with the same functionality, where s_i $(1 \leq i \leq n)$ represent a web service.

3. Q_{m*n} is the user-service invocation matrix, and $q_{ij} \in R$ shows a value of QoS that the service *j* has for user *i*.

The number of user-service matrices that represent the values of QoS attributes given to web services by users will be the number of QoS attributes. Figure (1.1) shows an example of a user-service matrix for the QoS values of response time and missing value prediction. This matrix shows the response time of each service for each user.

When a user uses a service, the response time or throughput can be calculated through a middleware. However, as the number of web services is very high, each user uses only a small number of services, thus, many entries of these matrices are empty and have no values. Therefore, the prediction problem is to predict the values of the empty entries of this matrix using the existing values of the matrix after correcting the unreliable data.



	S1	S2	S3	S4	S5
u1	0.45		0.3		
u2		0.73		0.32	
u3		0.55	0.3		0.50
u4	0.4			0.46	

User-Services Matrix

	S1	S2	S3	S4	S5
u1	0.45	0.5	0.3	0.38	0.5
u2	0.3	0.73	0.33	0.32	0.55
u3	0.4	0.55	0.3	0.4	0.50
u4	0.4	0.35	0.30	0.46	0.45

Predicated User-Service Matrix

Figure (1.1) Example of Predicting User-service Matrix

1.4 Objectives of the Dissertation

The objectives of this dissertation are to present a model for building a personalized prediction for the QoS of non-functional attributes after detecting untrusted users through user reputation analysis, from which, users and application designers can obtain a suitable service that efficiently satisfies both functional attributes and non-functional attributes of user requirements. The system uses a real-world dataset of service users and QoS values of service invocations by different users to use for the prediction process.

A user's reputation can be determined by analyzing its past observations to identify untrusted users and correct their unreliable data, followed by computing QoS prediction according to users' location information and reliable data.

In this dissertation, a CF -based prediction approach is built by relying on multiple contextual information: User reputation as implicit context and user location information as explicit context to support the selection of optimal service for active user.

1.5 Dissertation Contribution

The main contributions of this Dissertation are:

1. A confidence aware prediction approach is presented that reduces the effect of unreliable QoS values to make a more accurate prediction and then fixed it to avoid data sparsity problem due to the immediate deleting of unreliable values.
2. Since the QoS values of services are affected by the geographical location of different users and the network environment, a hybrid approach is proposed that combines

CF-based technique with context-based method. In this proposed approach, the data volume is reduced by clustering the data so as to decline the computational size and mitigate the impact of data sparsity problem.

1.6 Related Works

Quality of Service (QoS) prediction for web services is widely discussed in the areas of service selection and service composition, and service recommendation in a dynamic environment [8]. In general, QoS prediction methods can be classified into three main categories: CF-based methods, context-based methods, and context-based CF methods that are the combination of Cf-based with context-based approaches. Furthermore, Collaborative filtering can be divided into main types: memory-based algorithms and model-based algorithms. Memory-based algorithms directly use the similarity between neighbors to predict the QoS values, while in model-based algorithms, the QoS values are predicted by the model created from the behavior of other users or services. [17].

Zheng et al. (2011) have presented a hybrid CF approach (UIPCC) User-Item-based Pearson Correlation Coefficient by combining user-based Pearson Correlation Coefficient (PCC) approach and item-based approach to make accurate prediction for QoS values, in which the similarity between users and services used to predict QoS values, and conducted a large-scale distributed QoS assessment of real-world services. This methods suffer from data sparsity and scalability problems [18].

Tang et al. (2012) have proposed a method for predicting QoS values. In this study, geographic information of the user and services is

combined with a memory-based CF method. This method shows that both user location and service location can improve QoS prediction. However, this method indicate that sufficient QoS entries are required for QoS prediction with normal performance [19].

Xu et al. (2015) have proposed a method called RAM based on user reputation and matrix factorization. Once the reputation of users is calculated, users are divided into different grades and then integrated into a matrix factorization (MF) prediction. However, This method may not produce efficient results if inappropriate parameters are selected [20].

Su et al. (2017) have proposed a trust-aware prediction approach called TAP, which uses K-Means algorithm and beta distribution method to compute users' reputation degrees and cluster honest users, and then identify a set of trusted similar users and services for predicting unknown QoS values. This approach mainly solve the problem of untrustworthy users on the accuracy of QoS prediction, but it do not solve the data sparseness problem [21].

Chen et al. (2017) have presented an approach called Geographical Neighborhood based Matrix Factorization (GNMF), this method is a combination of Matrix Factorization and clustering web services using a bottom-up hierarchical neighborhood clustering algorithm in which neighbors are clustered according to the provider, autonomous system (AS), and country, after clustering web services to determine neighborhood information, the similarity between services and their combination with the geographical location of the service provider is used to predict unknown QoS values. However, this study focus only on transferring knowledge from a web service' geographical neighborhood [22].

Zhu et al. (2018) proposed a location-aware method for prediction QoS that consider a user's privacy introduced as a problem of recommender system. It proposed a similarity-maintaining privacy presentation (SPP) strategy to overcome this problem and then used the L1-norm low-rank matrix factorization technique to predict QoS values. This technique used longitude and latitude of services as a location information to present a location-aware method. This method predicts the missing QoS values without considering whether the QoS values are reliable or not [23].

Chen et al. (2019) presented a location-based prediction method (LANFM) that combines the neural networks with a factorization machine to predict the QoS values of the web services. In this method, the location information of the users and web services converted to embedding vector by a neural network, and then a factorization machine used to perform a prediction [24].

Chen et al. (2020) have developed a model called WRAMF to handle the wide-range change problem of QoS data. They could get more accuracy in predicting QoS values by training the model with parallel stochastic gradient descent algorithm [25].

Wang et al. (2020) proposed a trust-aware CF method is presented for predicting QoS values, which first identifies user trust by using a two-phase K-Means algorithm to exclude unreliable users. Then, reputation-based network embedding is used to learn the hidden representations of users. Finally, the user-based CF is used to predict the missing QoS values [26].

Hasnain et al. (2020) proposed an approach for ranking web services was proposed based on a trust score measuring. In this method, a

confusion matrix was used to determine the trust scores of all web services [27] .

Smahi et al. (2021) proposed a deep learning method for predicting QoS values that combines MF technique based on deep autoencoder and clustering to overcome the sparsity problem in CF. In this method, after clustering users and services using a self-organizing map according to their geographical location, a deep autoencoder is trained for each cluster, and finally unknown QoS values are predicted by the trained deep autoencoder with respect to the nearest cluster [28].

Based on previous research work, we can see that most approaches do not consider the both problems of reliable QoS preprocessing and data sparseness in QoS prediction. Therefore, we propose a personalized QoS prediction to solve the two challenges simultaneously based on reputation and location-aware collaborative filtering for web services. The proposed approach can be divided into three steps. In the first step, a set of untrusted users is identified using Dirichlet probability distribution and correct their contributed QoS values with the average value of the reliable QoS interval for each web service. In the second step, all users are clustered by geographic location using the DBSCAN clustering algorithm to improve the prediction accuracy. In the third step, the missing QoS values are predicted using a collaborative filtering approach based on the similarity between users in each cluster.

Abstracts of above previous work for other researchers have shown in Table 1.1.

Table (1.1): Summary of the Mentioned Related Works

No.	Reference	Technique	Dataset
1.	Zheng et al. (2010)	Memory-based	WSdream-dataset2
2.	Tang et al. (2012)	Memory-based & Location-aware	WSdream-dataset2
3.	Xu et al. (2015)	Reputation-aware & Matrix Factorization	WSdream-dataset2
4.	Su et al. (2017)	Trust-aware & Clustering	WSdream-dataset2
5.	Chen et al. (2017)	Matrix Factorization & clustering & Location-aware	WSdream-dataset2
6.	Zhu et al. (2018)	Matrix Factorization & Location-aware	WSdream-dataset2
7.	Chen et al. (2019)	Neural Network & Matrix Factorization & Location-aware	WSdream-dataset2
8.	Chen et al. (2020)	Matrix Factorization & Neural Network	WSdream-dataset2
9.	Wang et al. (2020)	Trust-aware & Clustering	WSdream-dataset2
10.	Hasnain et al. (2020)	Trust-aware	WSdream-dataset2
11.	Smahi et al. (2021)	Deep learning & Matrix Factorization & Clustering	WSdream-dataset2 WSdream-dataset3

1.7 Dissertation Outline

This dissertation contains five chapters. It is organized as follows: -

Chapter Two reviews Cloud Computing, Service Oriented Architecture (SOA), Quality of Services, Recommendation Systems, Clustering algorithm, Reputation and Trust system.

Chapter Three provides the details of the proposed system, which includes the block diagram and the algorithms to implement the system.

Chapter Four discusses the experimental results of the proposed system and compares with other works.

Chapter Five explains the conclusions of the dissertation, and suggestions for future work.

Chapter Two

Theoretical Background

2.1 Introduction

In recent years, most computer software has transformed from native software to Web-based software, and technologies such as software Oriented Architecture (SOA), Internet of services (IoS), and cloud computing have found popularity[10]. However, the services provided on cloud computing grown dramatically, resulting in a multiple of services with similar functionality, making it difficult for consumers to select the best service. Consumers consider Quality of service (QoS) of the web services to select the best service from among them. The prediction of QoS values of the web services and recommendations of the best service based on these values to the consumers is one of the major challenges in the web service area [29]. A suggested way of solving this challenge is through using Recommender Systems (RSs). The initial goal of these systems is to provide the user with service and data recommendations that may interest them based on these demands and tendencies. Recently, RSs have increased in popularity and are used in different topics. This leads to the notion that an RS could possibly be an essential step in helping users identify the service of interest, thus solving the challenge of selecting the best services [30].

2.2 Cloud Computing

Cloud computing is attracting a lot of interest and is being used on a large scale recently [31]. It has evolved into a scalable platform for the use and delivery of services. One of the technical foundations of cloud computing is Service-Oriented Architecture (SOA), where services from different cloud providers are found and integrated over the Internet. SOA is becoming a popular and important framework for building web applications in the era of Web 2.0 [32].

Cloud computing provides shared resources (e.g., infrastructure, platform, and software) as services available over the network [33]. It is used in a variety of applications, including academic, business, government, and many others, not only for cost saving, but also to achieve strategic IT and business goals. However, Cloud services are used where and when needed, and in the amount needed, and only pay for the resources that they actually use [10].

The important characteristics and their description are displayed in Table (2.1) [34].

Table (2.1) Cloud Characteristics.

Cloud Characteristic	Description
On-demand self-service	Computing capabilities (e.g., server time and network storage) can be unilaterally automatically provisioned as needed).
Broad network access	Capabilities are accessible through heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
Resource pooling	Computing resources (e.g., storage, processing, memory, and bandwidth) are pooled to serve multiple consumers, and are dynamically assigned and reassigned according to demand. Customers have no control over the exact location of resources, but may be able to specify location (e.g., country, state, or datacenter).
Rapid elasticity	Capabilities can be elastically provisioned and released commensurate with demand. Available capabilities often appear to be unlimited.
Measured service	Resource use is automatically controlled and optimized through metering capabilities, appropriate to type of service (e.g., storage, processing, bandwidth, and active user accounts).
Multitenancy	Cloud computing is a shared resource that draws on resource pooling as an important feature. It implies use of same resources by multiple consumers, called tenants.

2.2.1 Entities Involved in Cloud-Based Services

The National Institute of Standards and Technology (NIST) has defined a reference architecture for cloud services. The reference architecture defines five unique actors involved in cloud- based services[35]:

1. **Cloud consumer:** A person or entity that uses the services offered by one or more cloud providers.
2. **Cloud provider:** An entity that provides services to cloud consumers under an appropriate deployment model.
3. **Cloud carrier:** One or more entities that provide network services between cloud providers and cloud consumers.
4. **Cloud broker:** An entity that negotiates relationships between cloud providers and cloud consumers.
5. **Cloud auditor:** An entity that can perform an independent verification of the cloud services offered and their usage.

2.2.2 Cloud Services Types

Network resources, computing resources, and applications offered as services to users by major providers. Cloud services can be divided into three main categories based on the type of services that are provided by the cloud provider: "Software as a Service" (SaaS), "Platform as a Service" (PaaS), and "Infrastructure as a Service" (IaaS) [36].

1. Software as a Service (SaaS)

Software as a Service (SaaS) is a delivery model for software applications over the Internet, typically on a subscription basis. It alleviates the need for the users to install and run the applications locally

on their computers, relieving them of the responsibility of repairing and maintaining the hardware and software. However, SaaS replaces traditional software usage with a subscription/rental model, reducing the cost of provisioning and managing the user's physical devices. Google Apps is an example SaaS implementation. It offers a number of web applications with similar functionality to traditional office applications.

2. Platform as a Service (PaaS)

Platform as a Service (PaaS) cloud computing models provide an environment for the execution of application services. A service provider hosts the tools, software, and platform for application development and makes them available to application developers so that they can easily program and deploy their application without having to interact directly with the underlying infrastructure. Microsoft Azure and Google App Engine are an examples of PaaS implementation.

3. Infrastructure as a service (IaaS)

In the IaaS model, essential computing infrastructure, such as network equipment, servers, storage, is provided as a service on demand. Customers get these resources as an outsourced service for as long as they need them instead of purchasing them. Amazon EC2 is an example IaaS implementation. In general, IaaS is divided into two categories:

1. **Computation as a Service (CaaS),** where virtual machine-based servers are rented and billed per hour based on virtual machine capacity
2. **Data as a Service (DaaS),** which uses unlimited storage space to store the user's data regardless of its type, billed per Gigabyte for data size and data transfer.

Table (2.2) shows cloud service types and their capability offered to the user.

Table (2.2) Service Type in Cloud Computing

Service Type	Capability Offered to the user
Software as a Service **(SaaS)**	Use of applications that run on the cloud.
Platform as a Service **(PaaS)**	Deployment of applications on the cloud infrastructure; may use supported programming languages, libraries, services, and tools.
Infrastructure as a Service **(IaaS)**	Provisioning of processing, storage, networks, etc.; may deploy and run operating systems, applications, etc.

2.2.3 Cloud Computing Types

Cloud computing can be categorized based on who owns and operates the cloud. Main types of cloud computing are public, private, and hybrid clouds [37].

1. **Public Cloud**, also referred to as external cloud, is the most common type of cloud computing, where services are made available to the general public on a pay-as-you-go basis.
2. **Private Cloud,** also referred to as internal cloud, is created, deployed, and managed behind the enterprise firewall for the exclusive use of the enterprise. The majority of private clouds are used by large corporations or government agencies that prefer a more controlled and secure environment for their data.
3. **Hybrid Cloud** is a combination of the above two types (private cloud and public cloud). A hybrid cloud allows an organization to keep its critical data and applications behind

its own firewall while provisioning less critical ones in the public cloud.

Figure (2.1) shows the main types of cloud computing.

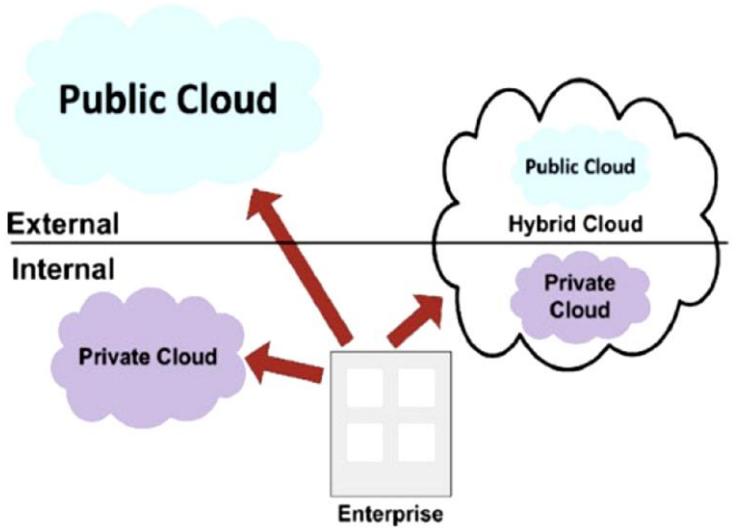


Figure (2.1) Cloud Computing Types

2.3 Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) and cloud computing are related to each other. SOA is an architectural pattern that defines a way to make computing components reusable and interoperable via service interfaces. This interface is a service contract between the service provider and the service user, while cloud computing is a set of enabling technologies that services a bigger, more flexible platform for enterprises to build their SOA solutions. In other words, SOA and cloud computing coexist, complement and support each other [31].

In recent years, web services technology has become the most popular and widely used technology for building SOA applications. Web services are based on a number of protocols and standards such as SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description, Discovery, and

Integration) [38]. Web services are distributed components that are self-contained, discoverable, reusable, composable, and have a transparent location. Due to its popularity, an increasing number of functionally similar web services can be found on the cloud computing, leaving the user to ask the question, "Which are the better services?" or "Which of them fits my needs better?". Users have the difficult task of selecting a suitable service for their requirements [39]. Quality of Service (QoS) has proven to be the most appropriate criterion to distinguish non-functional features between equivalent web services [40].

In SOA, there are three main roles: the service provider, the service user, and the service broker, as shown in Figure (2.2). The service provider defines a service in a WSDL file. This is published to the service broker using UDDI so that the service is discoverable by the service user. The service user requests the service from the service broker. The service broker makes the WSDL file available, and the service user consumes the service directly using SOAP [41].

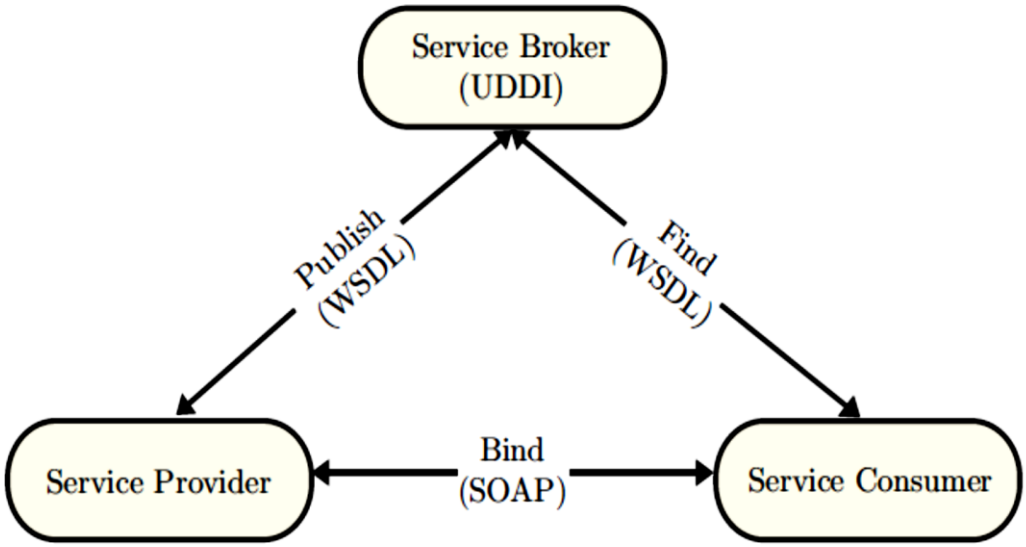


Figure (2.2) SOA Structure

2.4 Quality of Services (QoS)

Quality-of-Service (QoS) is widely employed to represent the non-functional performance of services and has been considered as the key factor to differentiate the qualities of service candidates. It becomes important to evaluate the QoS performance of web services. Therefore, determining the QoS values of web services is a necessity, and most of the QoS attributes have dynamic characteristics, and the values of them are different for every user, and variable in time for a specific user [42].

For the first time in 2003, Ran introduce a service discovery method that considered the non-functional attributes of the Web services and introduced QoS as an important challenge in the Web service domain. Ran defines the quality of services as a set of non-functional attributes that may impact the quality of services offered by a Web service [12]. In another definition, the QoS of Web service defined as "a set of non-functional attributes of the entities used in the path from the WS to the client that bear on the WS' ability to satisfied stated or implied needs in an end-to-end fashion" [43].

Given the importance of QoS in Web services, two issues must be identified about these features:

(1) What are the QoS attributes of the services?

(2) How should these values be determined?

To specify the QoS features, various models are provided for the quality of Web services, and in different researches, metrics defined for Quality-of-Service evaluation [12], [44], [45]. Oriol et al. have systematically compared these models and recognized the QoS attributes of Web service [46]. The most important QoS attributes of Web services

and their definition are displayed in Table (2.٣). Tao divides the QoS attributes into two categories of objective and subjective attributes [47].

Table (2.3) QoS Attributes Definition of Web Services

QoS Attribute	Definition
Accuracy	Error rate produced by the service.
Accessibility	Degree the service is capable of serving a Web service request.
Capacity	Limit of concurrent requests for guaranteed performance.
Response Time	Time to complete a Web service request (from a client perspective).
Throughput	Number of Web service requests served at a given time period.
Availability	The probability that the service can respond to the consumer requests.
MTTR	Meantime to repair.
Interoperability	The ease with which a consumer application or agent interoperates with a service.
Robustness	The degree to which a service can function correctly in the presence of invalid incomplete or conflicting inputs.
Authentication	A measure of how the service authentication principals who can access services and data.
Confidentiality	A measure of how the service threat the data, so that only authorized principals can access or modify the data.
Cost	Cost involved in requesting a service.
Reputation	Measure the user rating to the service.

subjective and rest are objective. Subjective characteristics are known as attributes obtained from the user's subjective evaluation of an items and depend on the morale, interests, and other subjective factors of the user. It considers objective features like the ones not explicitly related to the user's interests. According to this definition, the main QoS attributes of services are in the second category as all the qualities of the services depend somehow on its implementation and the quality of its

construction. For example, reputation, as a subjective parameter, depends on the user' satisfaction with the service provided, and the satisfaction refers to parameters such as availability of the service, response time etc. which are also objective [48]. In another categorization, these attributes are divided into static and dynamic categories. Static QoS attributes are attributes whose value do not change, such as cost, and dynamic QoS attributes are the ones whose values depend on other parameters and can change [49]. Thus, one can say that most of QoS attributes are dynamic and objective.

In response to the second question, how should the values of QoS attributes be determined? Should be said, these features cannot be recorded as other functional features with the service provider's notice in the registry. Because (1) some service providers may announce some unrealistic values for these features. (2) Dynamic QoS features are dependent on the provider, the user, and the network. For instance, the response time of the service depends on the user's location, the time of the service invocation, and the network infrastructure [49]. Thus, the evaluation of services by users can obtain more accurate values for these features [50], [18] on the other hand, evaluating web services from the user's side, which is virtually impossible and has following problems [18], [51]:

1. Using all services needs a lot of time, and some services need a fee that making the process costly.
2. As the number of Web services is high and many Web services are added to these services, some of them may not be evaluated and practically not all services will be deployed.
3. Users are not well-informed enough to be able to measure all of these features.

Considering the above, there should be another way to determine QoS values to be both accurate and feasible. For the first time, Shao uses a collaborative filtering prediction method to determine the values of QoS attributes of Web services [28]. After that, many researchers have also worked on this topic. Thuse, predicting QoS values of services is one of the main approaches for determining the QoS values of Web services.

2.5 Recommendation Systems

There are a wide range of web applications that involve predicting options for users. These applications are called recommender systems (RS). In a recommendation system application, there are two classes of entities, which we shall refer to as users and items. Users have preferences or ratings for certain items, and these preferences or rating must be teased out of the data. The data itself is represented as a utility matrix, giving for each user-item pair [52]. However, Recommender systems collect information about users and items, which can be explicit, such as user ratings, or implicit, such as monitoring the behavior of users [53].

Recommender systems use a number of different techniques. We can divide these systems into three categories [52]:

1. Collaborative filtering technique recommend items based on similarity measures between users and/or items.
2. Context-based technique systems examine properties of the items recommended.
3. A hybrid approach that combines collaborative filtering and context-based techniques.

2.5.1 Collaborative Filtering Technique

One of the most important approaches for predicting user interests in commercial system is collaborative filtering. In these approaches, we focus on the similarity between users or items. The basic idea of CF is that if two users X and Y rate items similarly or exhibit the same behavior, their behavior or rating is similar to another item [54]. This method is widely used in commercial system such Amazon eBay, etc. [18]. For the first time, Shao et al. used the CF method to predict Web service QoS values. They, using similarity between users, predict QoS values of services for a particular user using QoS values of the same service for similar users [55].

In general, CF-based method can be classified into two main categories: memory-based Methods and model-based methods [56].

1. Memory-based Methods

In these algorithms, using a relatively simple statistical equation, QoS values for a user are calculated using the values of similar users or similar services that we know as neighbors. Memory-based or neighborhood-based approaches have two main steps for prediction of QoS values. The first step is similarity computation and the second is predicting the unknown QoS values by the value of similar user or service. One of the main steps in memory-based approaches is to determine the similarity between users or service. The most important similarity measurement method is Pearson Correlation Coefficient (PCC). This criterion has good performance when vectors are continues [57].

After calculated the similarity between users or the similarity between Web services, the next step is to predict QoS values of the services for their users. Usually, at this step, top-k users or services that

are most similar to the active users or serves are selected. Memory-based approaches include user-based, item-based and hybrid algorithms [58] .

A. User-based CF method:

In user-based method, by finding similar users to the active user we can predict the QoS values of services for the active user.

B. Service-based CF method:

In this method, the similarities between different items are calculated and then user rating to an item will be predicted by the rating values of the similar items.

C. Hybrid CF method:

In this method, both the similarities between services and the similarities between users are computed. Then, a hybrid algorithm uses these similarities to predict QoS values. These algorithms have a better result than previous algorithms.

Memory-based algorithms have acceptable accuracy when the user-service matrix is nearly complete, but these methods suffer from some problems [57]:

i. **Data sparsity:** given the high number of web services, users have only used very few numbers of these services, so the predicted QoS values do not have the required accuracy.

ii. **Cold-start:** another problem with the collaborative filtering methods is when a new service or new user is introduced to the system. Therefore, it is not possible to predict QoS values.

iii. **Scalability:** if the number of services and users is very high, then the cost of calculation of the neighbors of services will be

high. Therefore, the scalability problem will be one of the problems of this approach.

iv. **Trust:** In the Cf methods, prediction is made using QoS values provided by other users. Thus, the accuracy of the predicted values will strongly depend on the values contributed by other users. There are several reasons why the values contributed by other users are not correct, such as malicious attacks or comments by competitors, or unrealistic positive comments from the service provider. Hence, distrusted in QoS values contributed by users can be one of the major challenges of CF-based methods.

2. Model-based Methods

These algorithms are presented to solve problems with memory-based methods. In these methods, using a user-service invocation dataset, a pattern is designed, and then by learning from these training techniques such clustering, matrix factorization, time series, and machine learning techniques are among these algorithms [10].

A. Clustering Algorithms

There are many algorithms and criteria for clustering data or object. A group of model-based CF algorithms is methods that cluster users or services. These methods reduce data volumes by clustering the data, increasing scalability, and reducing the volume of computation. Additionally, the cold start problem is also somewhat solved [59], [60].

B. Matrix factorization Algorithms

Matrix Factorization (MF) is the most important model-based approach used for QoS prediction and known as a successful method in

prediction and recommender system. In this method, the user-service matrix $Q = \{q_{ij}\} \in R^{m*n}$ is decomposed to the user feature matrix $U \in R^{d*m}$and the service feature matrix $U \in R^{d*m}$, so the m is the number of users, n is the number of services and d is the number of latent factors. the U and S matrices can be obtained by learning process. The main advantage of this method is that it can somewhat solve the problem of data sparsity and scalability [61]. Despite the advantages, however, these methods have problems, the most important of which are the following:

i. If a user or a new service is added, the model must be recreated and trained. Therefore, the overhead of updating these methods is high.

ii. In most of these methods, there are some parameters that need to be set properly and their values have a significant impact on the prediction accuracy.

iii. These methods do not have the ability to explain the reason for recommending a service, which is a relatively important feature for recommender systems.

C. Machine Learning Techniques

Machine learning techniques are another category of model-based methods that have been used for QoS prediction [62]. In recent researches, machine learning techniques are used for prediction, given that these techniques could model the complex problems with many features well. The most important of these techniques used in predicting QoS values are:

1. Deep neural networks are the most sophisticated approach that can be used to overcome the limitations of matrix factorization. It allows modeling nonlinear interactions in

the data and finding hidden patterns in the data that otherwise could not be detected [63].

2. Some studies used combining fuzzy logic and neural networks to predict unknown QoS values [8].
3. Autoencoder is another prediction method that predicts the QoS values by using a stacked autoencoder, which is a type of neural network [64].

D. Time Series Approaches

Due to the dynamic status of the network and its variable traffic, some QoS attributes such as response time, availability or throughput at different times have different values. In such cases where the value of a property is time-dependent, time series can be a good tool for predicting values [49].

2.5.2 Context-based Technique

Another most commonly used in recommendation system is a context-based technique [52]. The context-based technique makes recommendations to the user based on the properties of items, relevant user information, and the interaction between them [65]. The main steps of context-based based filtering are [66]:

1. Extract the item attributes to create item profile for all items.

2. Create the user profile for each active user.

3. Compare the item profile with user profile.

4. Recommend the items that most closely match the user profile and are not seen by the user.

We can divide context information into three categories:

1. **Computing context:** such as routing information, network properties, hardware and software used.
2. **User context and Item context:** such as geographic location, profile information,
3. **Time context:** such as the time and date of service usage.

Contextual information can be obtained in two ways [67]:

1. Explicit acquisition through directly collected data using registration or rating modules.
2. Implicit acquisition through the analysis of data from service properties and user interactions.

In general, context-based methods can be divided into two main categories: pre-filtering method and post-filtering method [66].

A. **Pre-filtering method:** This method filters the dataset before applying recommendation algorithms.

B. **Post-filtering:** Here the whole dataset is directly used by recommendation algorithms.

2.5.3 Context-aware CF-based Methods

There have been a number of prediction algorithms for web services QoS that combine memory-based or model-based CF techniques with a context-based approach and provide a hybrid approach. In these methods, context information is used by the user or service which can contribute to improving prediction accuracy. Two important contexts employed in these methods are the service or user geographical data and the service invocation time. According to CF methods, the prediction is based on the contributed data from users, so the trustworthiness of this data is an important problem, and for some reason, some of these data may be unreliable. Thus, location, time and trust are three important

contextual factors in determining QoS values of services and increasing the accuracy of prediction [19], [68], [69] .

A. Location-Aware Methods

QoS values such as (response time, availability and throughput) highly dependent on the performance of underlying networks [61]. So, the users that located in the same location have similar QoS attributes values for the same service, because they have a similar distance to the service provider and similar infrastructure and users that located in different location observed different QoS values for the same services [70]. Thus, a category of algorithms, by adding the geographic location of the service or user, or both, in a CF-based manner, try to improve the accuracy of the prediction of QoS values.

B. Trust-Aware Methods

In CF-based prediction methods, new QoS values will be predicted based on the values published by other users. Thus, the accuracy and precision of these values will depend on the accuracy and precision of the values published by the users [71]. Thuse, one of the problems with CF-based methods is invalid data. For avoiding this problem, service or user reputation is usually considered as a parameter. Reputation is a communication parameter derived from the experiences of using a service by the user, reflecting features such as reliability, credits, and integrity of services or user [72]. Moreover, a user's trust in another user or web service can also be used as a criterion to ensure the accuracy of the published data.

2.6 Similarity Measures

Similarity measures are also called similarity metrics. They are methods for calculating scores that express how similar users or items are

to each other. These scores can then be used as the basis for user-based or item-based recommendations. Similarity values can be numeric in the range [-1,1], where the positive value indicates the similarity and the negative difference. Depending on the context of use, similarity metrics can also be referred to as correlation metrics or distance metrics [52].

There are several methods for measuring the similarity between vectors [73] [74] [75]:

1. Jaccard Similarity

The Jaccard similarity of sets is the ratio of the size of the intersection of the sets to the size of the union. This similarity measure is suitable for many applications. For example, if the utility matrix reflects only purchases, this measure would be a good choice. However, when the utility matrices are more detailed ratings, the Jaccard distance loses important information. The equation used in this method of calculating similarity between two users is as follows:

$$\boldsymbol{sim(u_a, u_b) = \frac{I_{u_a} \cap I_{u_b}}{I_{u_a} \cup I_{u_b}}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

Where I_{u_a} and I_{u_b} represent rate values of user a and user b for items, respectively.

2. Euclidean Distances

One of the most important measures of similarity is the Euclidean distance. An *n*-dimensional Euclidean space is a space in which points are vectors of *n* real numbers. The conventional distance measure in this space, called the *L_2-norm*, is defined:

$$\boldsymbol{d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

The generalized form of Euclidean distance is termed as Minkowski Distance. For any constant *r*, we can define the L_r*-norm* to be the distance measured defined by:

$$\boldsymbol{d(x,y) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r}} \quad \text{..... (2.3)}$$

The case *r=2* is the usual L_2*- norm*.

Another common distance measure is the L_1*-norm* or Manhattan distance and is defined as the following equation:

$$\boldsymbol{d(x,y) = \sum_{i=1}^{n} |x_i - y_i|} \quad \text{..... (2.4)}$$

After calculating the distance, similarity is calculated as follows:

$$\boldsymbol{sim(x,y) = \frac{1}{1+d(x,y)}} \quad \text{..... (2.5)}$$

3. Pearson Correlation Coefficient (PCC)

The similarity between any two vectors can also be calculated using Pearson Correlation. It is used to measure the extent to which two variables linearly relate with each other. This criterion has good performance when vectors are continuous. The equation used in this method to compute the similarity between users is as follows:

$$\boldsymbol{sim(u,v) = \frac{\sum_{i \in I}(r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v})}{\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r_u})^2} \sqrt{\sum_{i \in I}(r_{v,i} - \overline{r_v})^2}}} \quad \text{..... (2.6)}$$

Where *u* and *v* are users of the services, the set $I = I_u \cap I_v$ represents the services that both *u* and *v* users have used and have values in the invocation matrix. $r_{u,i}$ is the rate is given by user *u* to the service *i*, and $\overline{r_u}$ is the mean of user's rating u to the various services.

For service similarity computing, PCC used the following equation:

$$sim(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i)\,(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2}\ \sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_j)^2}} \quad \ldots\ldots (2.7)$$

Where the set $U = U_i \cap U_j$ shows the users who used both service *i* and service *j* in the past and $\bar{r}_i$ is the average of the rate given to service *i*.

4. Cosine Similarity Measure

Measuring cosine similarity is more about the orientation of the two points in space than their exact distance from each other. Therefore, the cosine similarity between the two points is simply the cosine of that angle. If two points were *90* degrees apart, their cosine similarity would be zero because *cos(90)=0*, and if two points were *0* degrees apart, that is, if they were on the same line, their cosine similarity would be *1* because *cos(0)=1*.

Given two vectors *x* and *y*, the cosine of the angle between them is the dot product of the vectors and divided by the lengths of the vectors and defined as the following:

$$sim(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.8)$$

2.7 Clustering Algorithms

Clustering refers to a process where the same data is placed in cluster and data is not similar to the other cluster. Therefore, clustering algorithms look for similarities or dissimilarities among data points. Clustering is an unsupervised learning method so there is no label associated with data points. The algorithm tries to find the underlying structure of the data [54].

The clustering algorithm is used in many QoS prediction methods and service recommendation systems that combine CF -based approaches by clustering users and services according to their geographical location. These algorithms achieved better accuracy in prediction by defining the user and the services region concepts. Also, they could extend scalability by reducing the time complexity of prediction [76].

As with all clustering methods for predicting QoS values of web services, there are two main features:

1. Clustering is only used for data analysis before predicting the original method, so it is always used in combination.
2. The clustering criterion in almost all methods is the geographical similarity of the users or services.

As a result, Clustering with improves the scalability of predictive methods by reducing the data space of services and users. Moreover, clustering with similar services or similar users can reduce the impact of data sparsity [76].

There are several types of clustering algorithms [77] [78] [79]:

1. Hierarchical algorithms create hierarchical analytics for a dataset. In these algorithms, each data point is initially treated as a single cluster. At each iteration, the hierarchical pattern could be subclassified as either a conglomerate or a divider, depending on the shape of the destruction.

2. Partitioning algorithms create k-partitions for a given dataset, as each partition represents a particular group. k-means and k-medoids are the two best-known examples of this type of clustering algorithm. In k-means, each group represents an

average of the centroid or data points within the cluster, while in k-medoids, the groups are represented by the medoids of the cluster, which are the data points located near the cluster center.

3. Density-based algorithms create clusters according to density, as they could identify arbitrarily shaped clusters. The basic idea of these algorithms is that the data which are in the high-density region of the data space are assigned to the same cluster. However, the given clusters are developed further whenever the number of objects or data points within the neighboring area exceed some threshold. DBSCAN algorithm is a well-known example of such an algorithm that forms clusters based on density connectivity analysis.

4. Grid-based algorithms, instead of applying clustering directly to data objects, but rather place data objects into grid cells by partitioning each dimension into a finite number of cells of equal length. Then, some statistical information about the objects in each cell is computed.

5. Model-based algorithms, the basic idea in these algorithms is to select a particular model for each cluster and find the best fit for that model. There are mainly two types of model-based clustering algorithms, one is based on statistical learning method and the other is based on neural network learning method.

Figure (2.3) shows clustering algorithms types [79].

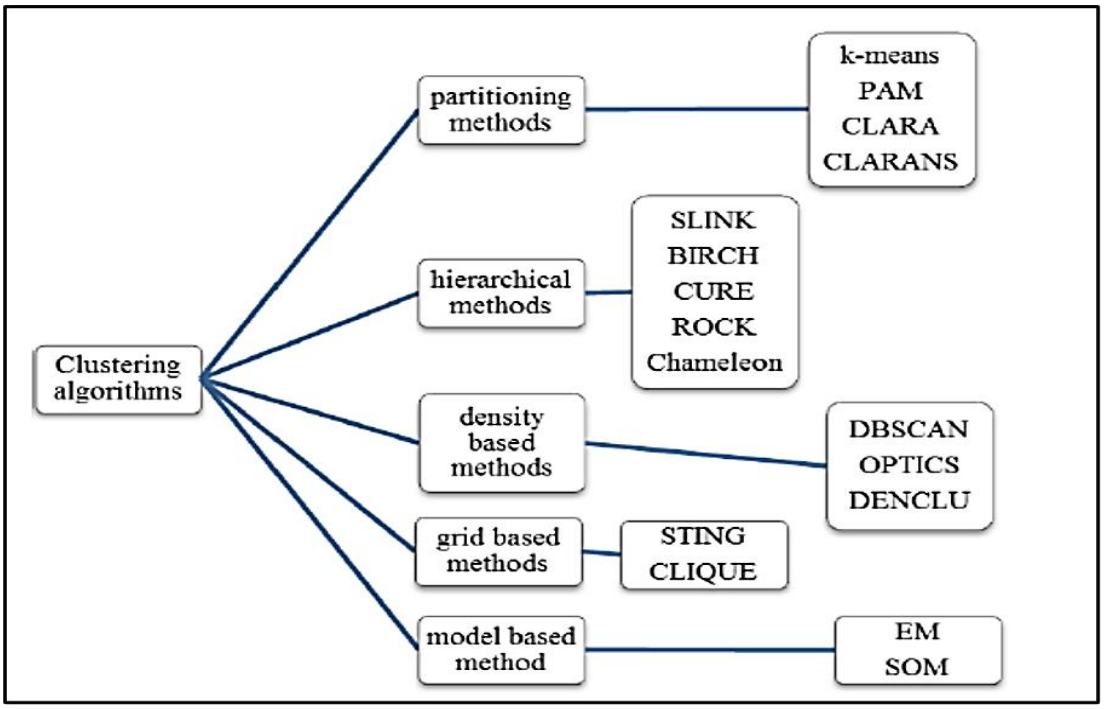


Figure (2.3) Classification of Clustering Algorithms

DBSCAN Clustering

DBSCAN (Density-Based Spatial Clustering of Application with Noise) is a technique for clustering data points into a number of partitioned clusters. DBSCAN is a type of density-based clustering approach that can identify arbitrarily shaped clusters, where dense regions are separated from lower-density regions to form clusters. The given clusters are developed further if the number of objects or data points within the adjacent area exceeds a predefined threshold [80].

There are two input parameters of DBSCAN are used to cluster data points [80]:

1. *MinPts*: Which represents a minimum number of points that must be included in a given cluster.
2. *Eps*: Which represents the radius of the cluster and is used to measure the distance between neighboring data points.

Two points are considered to be neighbors if the distance between them is less than or equal to *Eps*.

Based on these two input parameters, points are classified as core points, border points, or noise (outlier):

1. **Core point:** A point is a core point if there are at least *minPts* number of points (including the point itself) in its surrounding area with radius *Eps*.
2. **Border point:** A point is a border point if it can be reached from a core point and there are less than *minPts* number of points in its surrounding area.
3. **Noise (Outlier) point:** A point is a noise if it is not a core point and cannot be reached from any core point.

Figure (2.4) shows the types of points (core, border, outlier) in the DBSCAN algorithm, where A and C points are core points, B is border point, points N are noise points, the arrow represent *Eps*, *minpts*=4.

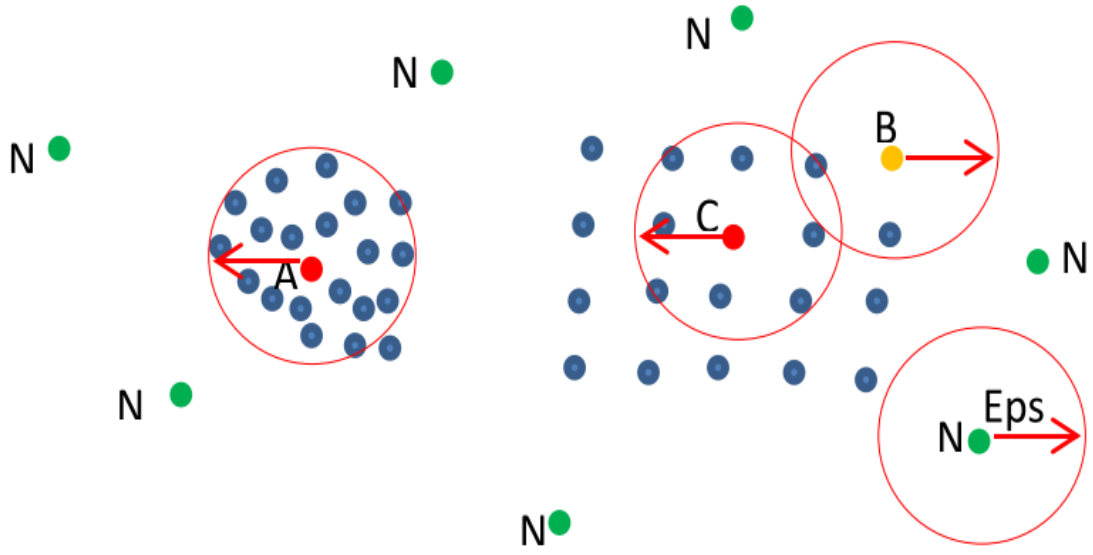


Figure (2.4) Types of Points in the DBSCAN Algorithm

The DBSCAN clustering algorithm is based on the following basic definitions [79]:

1. *Eps*-neighborhood of a point *x* denoted by $N_{Eps}(x) = \{y \in \text{Dataset} \mid dis(x,y) \leq Eps\}$.
2. Point *x* is directly density reachable from point *y* with respect to Eps and *minpts* if:
   a. $x \in N_{Eps}(y)$.
   b. $|N_{Eps}(y)| \geq minpts$
3. Point *x* is density reachable from *y* with respect to *Eps* and *minpts* if there exist a chain of points $z_1, z_2, \ldots, z_n$. Where $y=z_1$ and $x=z_n$, such that each point in the chain is direct density reachable from the previous one as shown in figure (2.5-a);
4. Point *x* is density connected to *y* with respect to *Eps* and *minpts* if there exist point *z* such that *x* and *y* are density reachable from *z* with respect to *Eps* and *minpts* as shown in figure (2.5-b) .
5. Cluster is non-empty subset of the input dataset with maximality and connectivity:
   a. If $x \in C$ and *y* is density reachable from *x* with respect to *Eps* and *minpts* then $y \in C$.
   b. If $x \in C$ and $y \in C$ then *x* is density connected to *y* with respect to *Eps* and *minpts*.
6. Noise is a set of points that are not belonging to any cluster. However, noise point does not belong to any neighborhood radius of core point and have number of points in its *Eps* radius less than *minpts*.

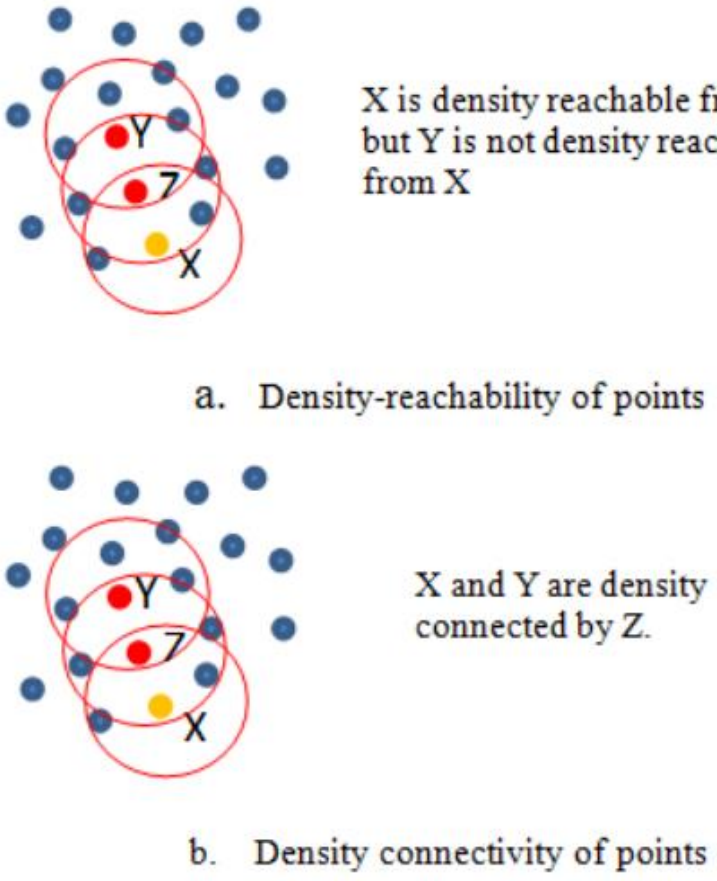


Figure (2.5) Density Reachability and Connectivity

The DBSCAN algorithm starts by picking up arbitrary point in the dataset that has not yet been visited, and checks the neighborhood based on a given value of Eps. If there is at least a minimum number of data points (*MinPts*) with an (*Eps*) to the current point, then all these points become part of the same cluster. Otherwise, that point will be marked as noise[79].

Most clustering methods use distance as a measure of the distance between two clusters, which fails to detect arbitrarily shaped clusters. DBSACAN can detect arbitrarily shaped clusters, which is the main feature of this technique in identifying clusters [81]. Algorithm (2.1) explain DBSCAN clustering [79].

Algorithm 2.1: DBSCAN Algorithm

```
DBSCAN(data,Eps,minpts)
Clus_id = 0
FOR I = 1 to size of data
 IF data[i] is unclassified THEN
   IF (|NEps(data[i])| ≥ minpts) THEN
      Clus_id = clus_id+1
      Expand_cluster(data[i], Eps, minpts, clus_id)
   ENDIF
 ENDIF
NEXT i
ALL unclassified points in data are noise
End DBSCAN
Expand_cluster(data, data[i], Eps, minpts, clus_id)
      Seed = data[i].regionquery(data[i],Eps)
      Data[i] and unclassified points in seed are
      assigned to clus_id
      Seed.delete(data[i])
      While seed <> empty do
            Point = Seed.getfirst
            Neighbor = point.regioquery(point,Eps)
            IF Neighbor.size ≥ minpts THEN
               Append all unclassified points in neighbor
               To seed and assign them to clus_id
            ENDIF
            Seed.delete(point)
      End while
End Expand_cluster
```

In Figure (2.6) an example about of DBSCAN clustering.

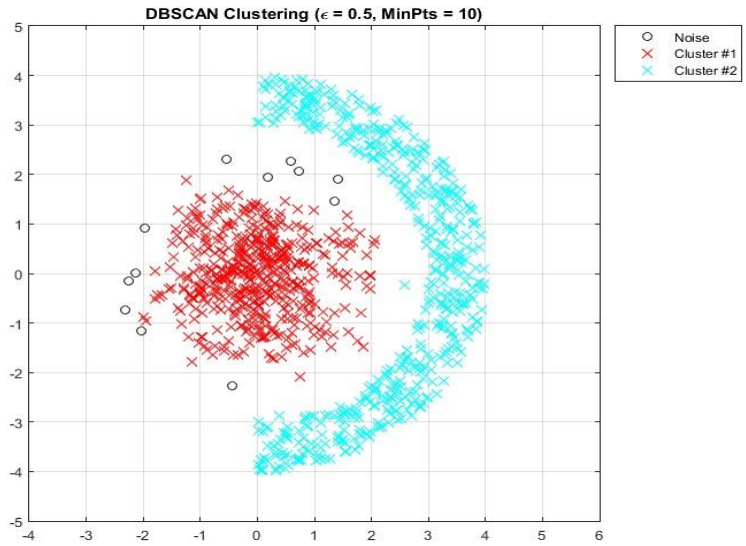


Figure (2.6) DBSCAN Clustering

2.8 Reputation and Trust System

On the Internet, users of services usually have inadequate information about the services offered, which can put users in vulnerable situations when purchasing these services. Thus, trust and reputation systems play an important role in decision making for Internet-mediated service delivery. The basic concept is that users submit their ratings about other users or for the services and then derive a reputation or trust score from the collected ratings about a particular user\service, which can help other users decide whether or not they want to deal with that user in the future [82].

The main differences between trust and reputation systems can be described as follows [83]:

1. Reputation can be viewed as a collective confidence measure in community which derived from member referrals or ratings of its members, while the subjective trust of an individual can be obtained from a received referrals as well as personal experience.

2. Reputation systems generate reputation score of an entity from the point of view of the entire community (public), while, trust systems generate a score that reflects the subjective opinion of the user on the trustworthiness of an entity (individual).

3. Transitivity is an explicit component in trust systems, whereas reputation systems usually consider transitivity only implicitly.

4. Trust systems usually use subjective and general measures of (reliability) trust as inputs, while information or ratings about specific (and objective) events, such as transactions, are used as inputs in reputation systems.

There is a similarity between collaborative filtering techniques and reputation systems, as both gather ratings from users. On the other hand, there is an important difference between them: Reputation systems assume that some users provide unreliable data for some reason, while collaborative filtering techniques assume that all users are equal in terms of credit and the values provided by them also reliable [83], [84].

2.8.1 Reputation Computation Methods

There are many methods for calculating reputation scores of entities in the Internet environment:

1. Traditional Computation Methods

In these methods, using a relatively simple statistical equations, In the simplest way, the reputation score can be calculated by summing up the number of positive and negative feedbacks separately and derive the total score as the positive feedbacks minus the negative feedback, an example of such a method is eBay's reputation system [85]. Another

method is to calculate the reputation score as the mean of all feedback, an example of such a method is Amazon and Epinions [86]. The main advantage of these methods is that they are easy to implement.

2. Dirichlet Reputation System

Dirichlet probability distribution, which is a multinomial Bayesian probability distribution. Dirichlet reputation systems represent a generalization of the binomial Beta reputation system. The multinomial aspect of the Dirichlet reputation system means that any set of discrete rating levels can be defined. This provides great flexibility and ease of use, as well as a solid foundation for designing reputation systems [87].

Multinomial Bayesian systems are based on computing reputation scores by statistically updating Dirichlet PDF. The a posteriori (i.e., updated) reputation score is calculated by combining the a priori (i.e., previous) reputation score with the new rating [88].

The probability distribution over disjoint elements of the multivariate state space are determined by the Dirichlet distribution, which describes the probability distribution over k-component random variable $p(\theta_i), i = 1 \dots k$ with sample space $[0,1]^k$, subject to the simple additivity requirement $\sum_{i=1}^{k} p(\theta_i) = 1$. [88]

The Dirichlet distribution captures a sequence of observations of the k possible outcomes with k positive real parameters $\alpha(\theta_i), i = 1 \dots k$ each corresponding to one of the possible outcomes. In order to have a compact notation, we define a vector $\vec{p} = \{p(\theta_i) \,|1 \leq\ i \leq k\}\ k$ to denote the k-component random probability variable and a vector $\vec{\alpha} = \{\alpha_i \,|1 \leq i \leq k\}$ to denote the k-component random observation variable $[\alpha(\theta_i)]_{i=1}^{k}$. Therefore, the Dirichlet probability function can be expressed as follows [88]:

$$\boldsymbol{f(\vec{p} \mid \vec{\alpha}) = \frac{\tau\left(\sum_{i=1}^{k} \alpha(\theta_i)\right)}{\prod_{i=1}^{k} \tau(\alpha(\theta_i))} \prod_{i=1}^{k} p(\theta_i)^{\alpha(\theta_i)-1}} \quad \ldots\ldots\ldots (2.9)$$

Where $\sum_{i=1}^{k} p(\theta_i) = 1, \quad p(\theta_1), \ldots, p(\theta_k) \geq 0$, $\alpha(\theta_1), \ldots, \alpha(\theta_k) > 0$, and τ is a gamma function subjective to $\tau(t) = \int_0^{\infty} x^{t-1} e^{-x} dx$.

The probability expectation value of the Dirichlet distribution is given by:

$$E(p(\theta_i) \mid \vec{\alpha}) = \left(\frac{\alpha(\theta_i)}{\sum_{i=1}^{K} \alpha(\theta_i)}, \right) \ldots\ldots\ldots\ldots (2.10)$$

Because of the additivity requirement $\sum_{i=1}^{k} p(\theta_i) = 1$,the Dirichlet distribution has only $k - 1$ degrees of freedom. This means that knowing $k-1$ probability variables and their density uniquely determines the last probability variable and its density.

2.9 Evaluation Strategies

Prediction accuracy is the most important criterion for evaluating QoS prediction of web services in collaborative filtering approaches. Two metrics are used, the first metric is the Mean of Absolute Error (MAE) and is obtained from the following equation:

$$\boldsymbol{MAE = \frac{1}{N} \sum_{i,j} |\bar{q}_{ij} - q_{ij}|} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.11)$$

Where $\boldsymbol{q_{ij}}$ is the actual value and $\boldsymbol{\bar{q}_{ij}}$ is the predicted value of QoS property, and $\boldsymbol{N}$ is the number of predicted QoS values.

Another metric used to measure predictive accuracy is the Root Mean Squared Error RMSE, and the following equation is used to calculate it:

$$RMSE = \frac{\sqrt{\sum_{i,j}\left(\bar{q}_{ij} - q_{ij}\right)^2}}{N} \qquad \ldots\ldots\ldots\ldots\ldots (2.12)$$

The lower values of these two criteria, the more accurate prediction we will have.

Chapter Three

The Proposed System

3.1 Introduction

This chapter clarifies how the proposed system is designed and implemented so as to build a personalized prediction for the QoS of non-functional values. It includes a description of the dataset that has been used in testing the system. As for the system itself, it consists of two stages, namely computing user reputation, and predicting unknown QoS values.

3.2 General Proposed System Architecture

Services are provided as shared resources in cloud computing, such as databases, software, and servers. Recently, as cloud computing has become more popular, an exponentially increasing number of cloud-based applications have emerged. The software architecture used to deliver multiple cloud services in a cloud environment is typically a web service. In the real world, the QoS of cloud services is strongly affected by the network environment and the location of the users. Moreover, only a limited number of these services are invoked by the service users. Therefore, predicting the QoS values of cloud services becomes an urgent and critical challenge in cloud services.

In this work of dissertation, two factors of contextual information were considered in building a proposed CF-based approach for predicting unknown QoS values: User reputation and geographical location. User reputation was used to identify untrusted users and fix their unreliable data. This ensures that the data contributed by users is credible, as well as reducing the impact of the data sparsity problem by correcting unreliable data. While geographic location has been used to improve similarity computation to improve the accuracy of QoS prediction.

Figure (3.1) shows the main parts of proposed system of the proposed work for Cf-based approach based on reputation and location-based. The Input data part is responsible for extracting, transforming and loading information from the sources. The data is divided into two types, namely QoS data and user locations.

In the user reputation part, to identify untrusted users in the QoS-dataset, preprocessing of QoS data is performed on the QoS data matrix, and then user reputation calculation is implemented using Dirichlet reputation system. Finally, the unreliable QoS data is processed to create a new QoS matrix.

In the QoS prediction part, to determine the neighbors of the users, a clustering process has been implemented based on the geographical location of the users. Then, the similarities between neighboring users in each cluster using cosine similarity are calculated according to the new QoS matrix. Finally, unknown QoS values are predicted based on the similarity of neighboring users.

This work uses a real-word dataset proposed by Zheng at al.[89] and available as WS-Dream. It contains four matrices: a user-list matrix that contains context information about 339 users, a WS-list matrix that contains context information about 5825 web services, and two other matrices that contain QoS data of web services., one showing response time data and the other showing throughput data of 5825 web services used by 339 users. The dataset contains 1,974,675 records of response time and throughput attributes.

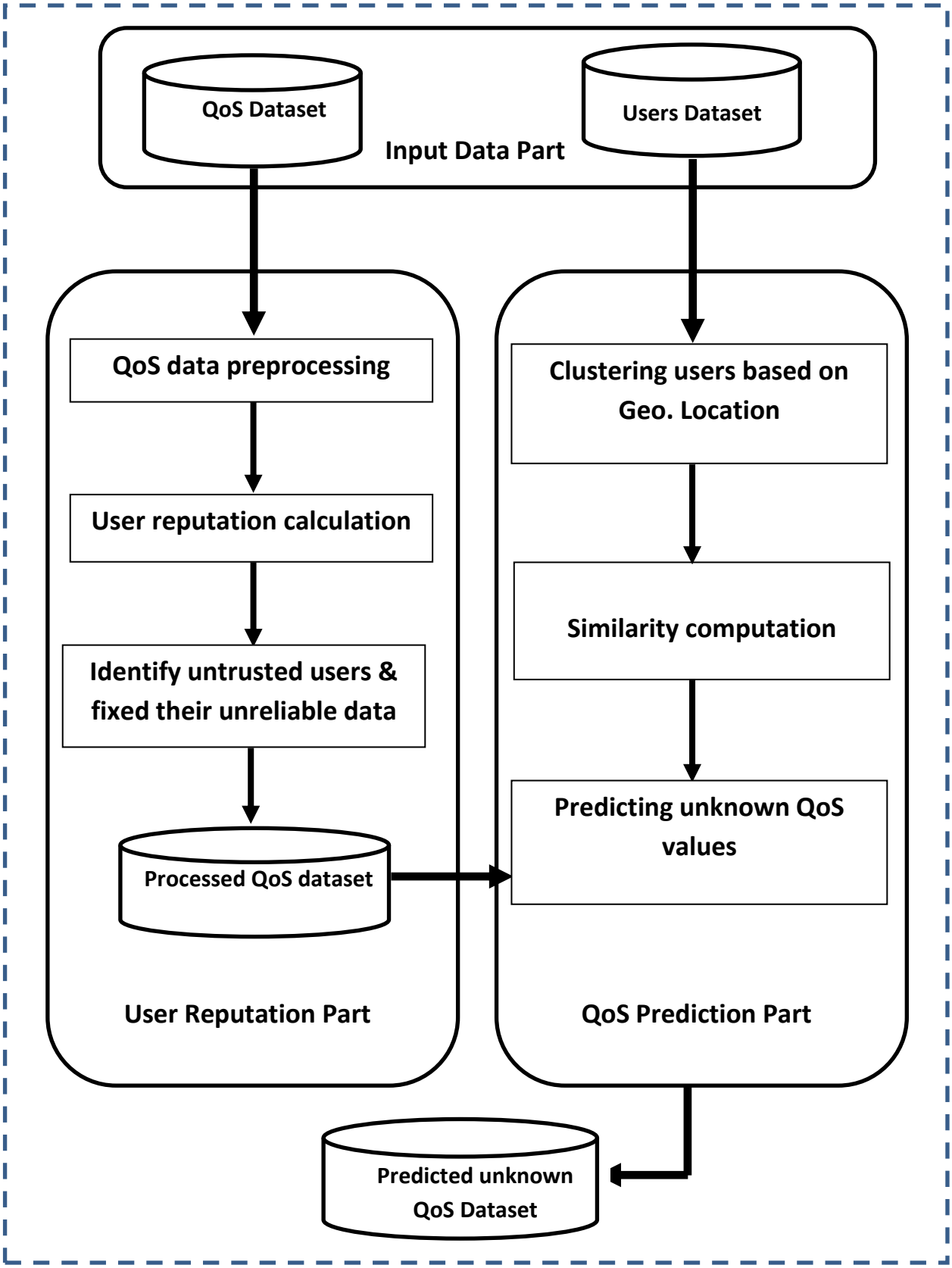


Figure (3.1) Proposed System Diagram

3.3 The Process of Personalized QoS Prediction

As shown in Figure (3.1), the process of the proposed approach for personalized QoS prediction consists of two main parts: 1) computing user' reputation part, where the QoS data obtained from different users is used to detect untrusted users. 2) predicting unknown QoS values part, where the missing QoS values are predicted by using reliable QoS values based on the similarity degree of all neighbors.

3.4 User Reputation-based Calculation

Non-functional QoS attributes are the ones whose values depend on other parameters and can change, including response time and throughput which are recorded and evaluated on the user-side. Therefore, the dynamic network environment leads to different QoS value observations made by different users while using the same services. For some reason, some of these data may be unreliable. Normally, the QoS values observations made by the majority of users should be part of the normal range, whereas any significantly deviating observations from this normal range are not likely to occur. Thus, the assumption is made that whenever users consistently submit QoS feedbacks that deviate significantly from the majority of users, then those users are not to be trusted. Consequently, the probability of the user's trustworthiness can be evaluated using user feedback information from historical invocation values.

3.4.1 QoS Data Pre-processing

The QoS data obtained during the invocation of web services by users are processed first. Since the QoS data is very different for all users of each web service, the QoS value of each service must be normalized to

a reasonable range to better partition the statistical intervals. The QoS values are normalized using the most common method of minimum and maximum normalization, often between 0 and 1. Each QoS value should be normalized as shown in the following Equation [90]:

$$NR_{ij} = \frac{R_{ij} - R_j^{min}}{R_j^{max} - R_j^{min}} \quad \text{..... (3.1)}$$

Where NR_{ij} is the normalized value of the original R_{ij}, R_j^{min} and R_j^{max} represent the minimum and maximum QoS values for service s_j, respectively. After normalization, all QoS values are scaled between [0, 1].

Algorithm (3.1) has shown how to normalize QoS value of QoS matrix.

Algorithm 3.1: Normalize QoS Values	
Input: QoS Dataset	
Output: Normalized QoS Dataset	
Begin	
1.	***for*** *j=1,2,3,...,M* ***do*** / M no. of services
2.	R_j^{min} ← ***Min value*** / Find minimum value of invocation for service j
3.	R_j^{max} ← ***Max value*** / Find maximum value of invocation. for service j
4.	***for*** *i=1,2,3,...,N* ***do*** / N no. of users
5.	NR_{ij} ← normalize QoS values by Eq. (3.1)
6.	*End for j*
7.	*End for j*
8.	**return NR** / Normalized QoS Dataset
End algorithm	

3.4.2 Determining Feedback Vector

In this section, the user feedback information is classified to trusted, untrusted, and no feedback. The feedback vector for each user in user-service matrix is obtained by first finding a set of users in each service that represent the reliable group, and then count the feedback information for each user based on trusted and untrusted feedback corresponding to the reliable group in each service. To find reliable user groups, the statistical interval division method is used to divide the normalized QoS data of all users for each web service within user-service matrix. For example, the normalized QoS value in [0, 1] intervals can be evenly divided into 11 intervals of (0, 0.1], (0.1, 0.2], ..., (0.9, 1], and [0, 0].

Since reliable users often represent the majority of users and their QoS data is within the normal range. Therefore, intervals with the majority of users are considered reliable. The user group in the reliable interval on the service s_j is defined as follows:

$$\boldsymbol{u_j^{max} = \{u | u \in G_j^l, l = argmax_k \left|G_j^k\right|} \quad \text{..... (3.2)}$$

Where $\left|\boldsymbol{G_j^k}\right|$ is the set of users in the kth groups, l represents the index of the group with the greatest number of users, $\boldsymbol{u_j^{max}}$ reflects the number of users in reliable group on the service s_j. As stated previously, QoS values that deviates highly from normal is unlikely, thus by assessing the QoS values introduced by the user and the deviation to the reliable group, the user feedbacks are classified into trusted and untrusted feedbacks. Trusted feedback refers to the QoS values is in the majority, otherwise it is untrusted feedback.

It is assumed that the quality of service (QoS) values for web services, contributed by various users, follow the Gaussian distribution $G(\mu, \sigma^2)$ with μ and σ being the average and standard deviation of reliable group in each service, respectively. Given the 3σ of Gaussian normal distribution [91], the probability that users observe QoS values in the range $[\mu - 3\sigma, \mu + 3\sigma]$ is very close to 99.7%, and thus the probability P of the QoS values of service s_j according to the observation of the user u_i is expressed as follows:

$$\boldsymbol{P\left(\mu_j^l - 3\sigma_j^l < NR_{ij} \leq \mu_j^l + 3\sigma_j^l\right) = 99.7\%)} \quad \text{…….. (3.3)}$$

Where μ_j^l and σ_j^l is the mean and standard deviation of the reliable group l of service s_j, respectively, and NR_{ij} is the normalized QoS value of service s_j observed by user u_i.

According to Equation 3.3, a normalized user feedback NR_{ij} can be classified as being either trusted, untrusted, or no feedback, as follows:

$$\boldsymbol{NR_{ij} = \begin{cases} Trusted\ feedback, & if\ \left|NR_{ij} - \mu_j^l\right| \leq 3\sigma_j^l \\ Untrusted\ feedback, & if\ \left|NR_{ij} - \mu_j^l\right| > 3\sigma_j^l \\ no\ feedback & if\ NR_{ij} = 0 \end{cases}} \quad \text{…….. (3.4)}$$

After classifying all user feedback information for each user, the feedback information can be counted and expressed as a feedback vector:

$$\overrightarrow{Fd_i} = [pt_i, pu_i, no_i] \quad \text{……..…. (3.5)}$$

Where $\overrightarrow{Fd_i}$ is the feedback vector of user i, pt_i is the number of trusted feedbacks of user i, pu_i is the number of untrusted feedbacks of user i, and no_i is the number of times without feedback.

Algorithm (3.2) has shown how to calculate users' feedback vector.

Algorithm 3.2: Calculate User's Feedback Vector

Input: **Normalizes QoS Dataset, Users list, Services List**

Output: **Feedback vector**

Begin

1.	***for*** *i=1,2,3,...,N* ***do*** / N no. of users
2.	$\boldsymbol{pt_i}$, $\boldsymbol{pu_i}$, $\boldsymbol{no_i}$ ← 0 / initialize Trusted, Untrusted and No Feedback values for user $_i$
3.	*End for i*
4.	***for*** *j=1,2,3,...,M* ***do*** / M no. of services
5.	divide normalized QoS values in each service into statistical intervals
6.	find the reliable interval with the maximum number of users according to Eq. (3.2)
7.	***for*** *i=1,2,3,...,N* ***do*** */ N no. of users*
8.	***if*** *no feedback for* $service_j$ *observed by* $user_i$ ***then***
9.	$no_i + 1 \leftarrow no_i$ / no_i no. of no feedback
10.	***else***
11.	find mean (μ) and standard deviation (σ) of reliable interval for $service_j$
12.	***if*** $\lvert NR_{ij} - \mu \rvert \leq 3\sigma$ ***then*** / NR_{ij} normalized QoS values
13.	pt_i+1 ← pt_i / pt_i no. of Trusted feedback
14.	**else**
15.	$pu_i + 1 \leftarrow pu_i$ */ pu_i no. of untrusted feedback*
16.	*End_if*
17.	*End_if*
18.	*End for i*
19.	*End for j*
20.	***for*** *i=1,2,3,...,N* ***do*** */ N no. of users*
21.	$\overrightarrow{\boldsymbol{Fd_i}} \leftarrow [pt_i, pu_i, no_i]$ / $\overrightarrow{Fd_i}$ Feedback vector for user $_i$
22.	*End for i*
23.	**return** $\overrightarrow{\boldsymbol{Fd}}$ / Feedback vector

End algorithm

3.4.3 Calculating User' Reputation

After determining the user feedback vector, the next step is to use the user reputation calculation to identify untrustworthy users. A user's reputation can be viewed as measure of trustworthiness of the community based on a user's historical feedback behaviour. Consequently, calculating a user's reputation can provide motivation for trustworthy feedback behaviour while helping users determine who is a credible user. In the proposed approach, the Dirichlet probability distribution is used for calculate a user's reputation based on the user's feedback vector.

As described earlier, suppose that the observed feedback $\overrightarrow{Fd_i}$ from user u_i contains pt_i trusted feedback, pu_i untrusted feedback, and no_i no feedback, therefore the vector $\vec{m} = (pt_i, pu_i, no_i)$ represents the count vector. According to the Dirichlet probability function, the feedbacks probability $\vec{p}$ of user u_i in the future can be expressed as follows:

$$\boldsymbol{f(\vec{p}|\vec{1}+\vec{m}) = \frac{\tau(pt_i+pu_i+no_i+3)}{\tau(pt_i+1)\,\tau(pu_i+1)\tau(no_i+1)} \cdot \left(p_{i,1}^{pt_i}\ p_{i,2}^{pu_i}\ p_{i,3}^{no_i}\right)} \quad \ldots. (3.6)$$

The probability expectation of the Dirichlet distribution is most likely the user's probability value $\vec{p}$. Thus, once the various user feedback is identified, the probability of trusted, untrusted, and non-feedbacks of future invocation services for the user u_i could be given by the following:

$$\boldsymbol{p_{i,1} = E(f(\vec{p}\,|\vec{1}+\vec{m}_i) = \frac{pt_i+1}{pt_i+pu_i+no_i+3}}$$

$$\boldsymbol{p_{i,2} = E(f(\vec{p}\,|\vec{1}+\vec{m}_i) = \frac{pu_i+1}{pt_i+pu_i+no_i+3}} \quad \ldots\ldots\ldots (3.7)$$

$$\boldsymbol{p_{i,3} = E(f(\vec{p}\,|\vec{1}+\vec{m}_i) = \frac{no_i+1}{pt_i+pu_i+no_i+3}}$$

Where $p_{i,1}$, $p_{i,2}$, and $p_{i,3}$ represent the probability of trusted, untrusted, and non-feedbacks for future invoking services by user u_i, respectively. The user's reputation is calculated only after the user invoking the service and receiving feedback. Thus, the user's reputation is considered to be the probability of trusted feedback whenever they receive feedbacks, and it can express as follows:

$$\boldsymbol{Rep_i = \frac{p_{i,1}}{p_{i,1}+p_{i,2}}, \quad (p_{i,1}+ p_{i,2}) \neq 0} \qquad \text{..... (3.8)}$$

Where Rep_i is the reputation of the user u_i, which is in the range of [0,1].

3.4.4 Identifying Untrusted Users and Fixed their Unreliable Data

After calculating user's reputation for all users according to the Dirichlet probability distribution and the QoS value of the web services observed by the user. The untrusted user is identified according to the user reputation value by setting a trusted threshold. If the user's reputation value is below the threshold δ, then the user is considered to be untrusted. The identification of untrusted users could be described in the following way:

$$\boldsymbol{\bar{u} = \{u | Rep_u < \delta\}} \qquad \text{..... (3.9)}$$

Where $\bar{u}$ represents the set of users identified as untrusted and δ represents the threshold. Obviously, if the unreliable data for untrusted users is removed in a direct manner, the user-service matrix becomes sparser, which greatly affects how accurate the QoS predictions are. To mitigate the sparse data problem, the unreliable QoS value submitted by untrusted users can be dealt with using the following equation:

$$r'_{u,j} = \begin{cases} \overline{Ru_j^{mean}} & , r_{u,j} > 0 \ and \ r_{uj} > \overline{Ru_j^{mean}} \\ 0 & , r_{u,j} = 0 \end{cases} \text{, and } u \in \bar{u} \quad \ldots. (3.10)$$

Where $r'_{u,j}$ is the modified QoS value of untrusted user u on service s_j, $r_{u,j}$ represent the original QoS value of that user u in service s_j, and $\overline{Ru_j^{mean}}$ represents the average QoS values in the reliable set of service s_j.

Algorithm (3.3) has explained how to calculate the user's reputation using Dirichlet Reputation system.

Algorithm 3.3: Calculate Users' Reputation		
Input: QoS Dataset, User's Feedback Vector		
Output: Reliable QoS Values		
Begin		
1.	***for*** *i=1,2,3,…..,N* ***do***	*/ N no. of users*
2.	$p_{i,1} \leftarrow \frac{pt_i+1}{pt_i+pu_i+no_i+3}$	/ $p_{i,1}$ the probabilities of trusted feedback of user u_i
3.	$p_{i,2} \leftarrow \frac{pu_i+1}{pt_i+pu_i+no_i+3}$	/ $p_{i,2}$ the probabilities of untrusted feedback of user u_i
	$p_{i,3} \leftarrow \frac{no_i+1}{pt_i+pu_i+no_i+3}$	/ $p_{i,2}$ the probabilities of no feedback of user u_i
4.	$Rep_i \leftarrow \frac{p_{i,1}}{p_{i,1}+p_{i2}}$	/ Rep_i the reputation of user u_i
5.	***If*** $Reb_i < \delta$ ***then***	/ δ threshold and equal to 0.5
6.	$\bar{u} \leftarrow u_i$	/ $\bar{u}$ untrusted users
7.	*End_if*	
8.	*End for i*	
9.	//Fixing unreliable QoS values	

10.	***for*** *u=1,2,3,….,Q* ***do*** / Q no. of untrusted users
11.	***for j =*** *1,2,3,….,M* ***do*** / M no. of services
12.	***If*** $r_{uj} > 0$ ***and*** $r_{uj} > Ru_j^{mean}$ ***then***
13.	$r'_{u,j} \leftarrow Ru_j^{mean}$ / Ru_j^{mean} mean of the QoS values in reliable interval of service j
14.	*End_if*
15.	*End for j*
16.	*End for u*
17	**return Reliable QoS values**
End algorithm	

3.5 Prediction Unknown QoS Values

QoS values are much determined by the performance of underlying networks. Therefore, the users located at the same location tend to present the same QoS attribute values on the service itself due to the similar infrastructures and the distances that they share to the service provider. Users with differing locations tend to observe different QoS values for the same services. Thus, adding the user's geographic location as context in a CF-based approach improves the accuracy of predicting missing QoS values and also mitigates the impact of data sparsity problem.

First, the DBSCAN clustering algorithm is applied to the user sides for forming clusters to reveal the geographical similarity between users. By clustering users according to their geographical location, the data volume is reduced and reduces the computational overhead. As mentioned before, the users in the same cluster tend to have QoS values that are alike.

There are two cases for predicting the unknown QoS value of service s_j for user u_i: (1) the service s_j was previously invoked by other

similar users in the same cluster. (2) the service s_j has not been invoked before by any user in the same cluster.

Algorithm (3.4) has shown how to predict the unknown QoS values.

Algorithm 3.4: Unknown QoS Values Prediction	
Input:	**U: User matrix** **R: Reliable QoS Matrix**
Output:	**R^*: Predicted Unknown QoS Values**
Begin	
1.	// Remove services that have not invocation by all users
2.	***for*** *i=1,2,3, …., M* ***do*** / M no. of services in the QoS matrix
3.	***if*** $S_i = \phi$ **then**
4.	*remove (S_i)*
5.	*End_if*
6.	*End for i*
7.	// Clustering users based on their geographical location
8.	C ← DBSCAN (minpts, Eps) / minpts: minimum no. of points /Eps: radius of the cluster
9.	// Predicting Unknown QoS Values
10.	***for*** *x=1,2,3, …., T1* ***do*** / T1 no. of clusters
11.	***for*** *i=1,2,3, …., T2 do* / T2 no. of users in cluster x
12.	***for*** *j=1,2,3, …., Q* ***do*** / Q no. of services that not previously invoked by the user i
13.	***if*** "there are neighbors who have previously invoked the service j" ***then***
14.	*predict R_{ij}^* in the same cluster x by Eq. (3.13)*
15.	*else predict R_{ij}^* in the closest cluster to cluster x by Eq. (3.16)*
16	*End_if*
17.	*End for j*
18.	*End for i*
19.	*End for x*
20.	**return** R^* **(Predicted Unknown QoS Values)**
End algorithm	

3.5.1 Predicting QoS Values in the Same Cluster

In the first case, the symbol φ_{ik} is used to refer to the set of neighbor users of user u_i in the same cluster which previously have observed QoS values for service s_k:

$$\boldsymbol{\varphi_{ik} = \{u_j | u_j \in C_n, R_{jk} > 0\}} \quad \text{..... (3.11)}$$

Where C_n represents the set of users within cluster n, R_{jk} denotes the QoS value of service s_k according to the observation of user u_j.

After all users in the dataset have been assigned clusters, the similarity between different users in each cluster can be calculated through the deployment of the cosine similarity. The following equation defines the similarity between different users:

$$\mathbf{Sim}(\boldsymbol{u_i, u_j}) = \frac{\sum_{k \in S_{ij}} \boldsymbol{R_{ik} R_{jk}}}{\sqrt{\sum_{k \in S_{ij}} (\boldsymbol{R_{ik}})^2} \sqrt{\sum_{k \in S_{ij}} (\boldsymbol{R_{jk}})^2}} \quad \text{..... (3.12)}$$

Where $Sim(u_i, u_j)$ denotes the similarity between user u_i and u_j. $S_{ij} = S_i \cap S_j$ represents the set of previously invoked services by both users u_i and u_j. R_{ik} represent the QoS value of service S_k according to the observation of user u_i. R_{jk} represent the QoS value of service S_k according to the observation of user u_j.

Then, the missing QoS value of service s_k for user u_i can be predicted by the following equation:

$$\boldsymbol{R_{ik} = \sum_{k \in \varphi_{ik}} \frac{Sim(u_i, u_j)}{\sum_{k \in \varphi_{ik}} Sim(u_i, u_j)}} \; R_{jk} \quad \text{..... (3.13)}$$

Where R_{ik} denotes the predicted value for s_k observed by u_i, R_{jk} denotes the QoS value of user u_j who has previously invoked service s_k.

Algorithm (3.5) shows predicting unknown QoS values in the same cluster.

Algorithm 3.5: Predicting Unknown QoS Values in the Same Cluster	
Input:	**C: User cluster** **R: Reliable QoS Matrix**
Output:	**R^* : Predicted Unknown QoS Values**
Begin	
1.	***for*** *i=1,2,3, …., N* ***do*** / N no. of users in cluster C that do not make previous invocation for some services.
2.	***for*** *k=1,2,3, …., M* ***do*** / M no. of services not invoked by user i.
3.	φ_{ik} ← *Construct user neighbor set for user i who invoked service k by Eq. (3.11)*
4.	***if*** $\varphi_{ik} \neq \emptyset$ *then*
5.	// calculate the similarity between users
6.	***for*** *j=1,2,3, …., T* ***do*** / T no. of users who have invoked services k.
7.	$\boldsymbol{Sim(u_i, u_j)}$ ← calculate the similarity by Eq. (3.12)
8.	*End for j*
9.	// predicting unknown QoS values
10.	***for*** *j=1,2,3, …., T do* / T no. of users who have invoked services k.
11.	$\boldsymbol{R^*_{ik}}$ ← predict the unknown QoS values by Eq. (3.13)
12.	*End for j*
13.	*End_if*
14.	*End for k*
15.	*End for i*
16.	**return R^*_{ij} (Predicted Unknown QoS Values)**
End algorithm	

3.5.2 Predicting QoS Values in the Closest Cluster

In the second case, if the neighbors of user u_i in the same cluster have not invoked service s_k before, the QoS values of user u_i can be observed in the closest neighbor cluster. Therefore, the distance between the user u_i and its cluster center c_i is first calculated with the centers of the other clusters. The distance between clusters can be obtained by the following equation:

$$\boldsymbol{d_{ij} = \frac{d(u_{i,}c_j)+d(c_i,c_j)}{2}} \quad \text{..... (3.14)}$$

Where $d(u_{i,}c_j)$ and $d(c_i, c_j)$ are the Euclidean distance between u_i and its cluster center c_i to the center c_j respectively.

The definition of the closest cluster for user u_i as d^{min} could be sated as follows:

$$\boldsymbol{d^{min} = d\left|d \epsilon D^i, i = argmin_k\right|D^k|} \quad \text{..... (3.15)}$$

Where $|D^k|$ denotes the distance of the kth cluster, i represent the index of the cluster with the smallest distance to the user u_i and its cluster center.

After assigning the closest group, the missing QoS value of user u_i for service s_k in cluster c is predicted according to the following equation:

$$\boldsymbol{R_{ik}^c = \frac{\sum_{K\epsilon\varphi_{ik}} R_{jk}^t}{N}} \quad \text{..... (3.16)}$$

Where R_{jk}^t denote to observed values of users who have invocation for service s_k in cluster t , N represent the number of users who have invocation for service s_k in the nearest cluster t.

If there are no users that have invoked the service s_k in the first closest cluster, the prediction of the QoS value of the service s_k observed by the user u_i is calculated in the next closest cluster.

Algorithm (3.6) shows predicting unknown QoS in the closest cluster.

Algorithm 3.6: Predicting Unknown QoS Values in Closest Cluster	
Input:	**C_s: Set of Clusters** **C : Current cluster** **R: Reliable QoS Matrix**
Output:	**R^c: Predicted Unknown QoS Values**
Begin	
1.	***for*** *i=1,2,3, …., N* ***do*** / N no. of users in cluster C that do not make Previous invocation for some services.
2.	***for*** *k=1,2,3, …., M* ***do*** / M no. of services not invoked by user i.
3.	φ_{ik} ← Construct user neighbor set for user i who invoked service k
4.	***if*** $\varphi_{ik} = \emptyset$ ***then***
5.	// calculate the distance between current cluster c_i for user i and other clusters
6.	***for*** *j=1,2,3, …., T1* ***do*** /T1 no. of clusters (C_s)
7.	c_i ← center of current cluster for user i
8.	c_j ← center of cluster j
9.	$\boldsymbol{d_{ij}}$ ← calculate the distance between location of user I and its cluster center j from another cluster by Eq.(3,14)
10.	*End for j*
11.	$t \leftarrow indexmin(d)$ / Find minimum distance
12.	φ_{ik}^t ← Construct user set for user i who invoked service k in closest cluster
13.	***if*** $\varphi_{ik}^t \neq \emptyset$ ***then***
14.	// predicting unknown QoS values in the closest cluster
15.	***for*** *j=1,2,3, …., T2* ***do*** T2 no. of users who have invoked services k in the closest cluster t
16.	$\boldsymbol{R_{ik}^c}$ ← predict the unknown QoS values by Eq. (3.16)

17.	*End for j*
18.	*End_if*
	End _if
19.	*End for k*
20.	*End for i*
21.	**return R^c (Predicted Unknown QoS Values)**
End algorithm	

Chapter Four

Implementation and Results

4.1 Introduction

In this section, we discuss the results of the experiment for the proposed approach using the real dataset to test the results and performance of the system available as WS-Dream that proposed and collected by Zheng et al [89]. The proposed system is applied on computer which meets particular criteria, among which are :

- Hardware, Processor: Intel(R) Core (TM) i7-8565U CPU @ 1.8GH_z (8CPUs). Memory: 8.00 GB RAM.
- Operating System: Windows 10 Pro 64-bit.
- Programming Language: Python 3.8

4.2 Dataset Description

In the real world, it is very expensive to invoke thousands of web services for large scale experiments. To evaluate our proposed QoS prediction approach, the experiments were conducted using the WS-Dream dataset proposed and collected by Zheng [89], which collects the QoS data of web service invocations and the context information of different users. Figure (4.1) shows how the dataset is collected.

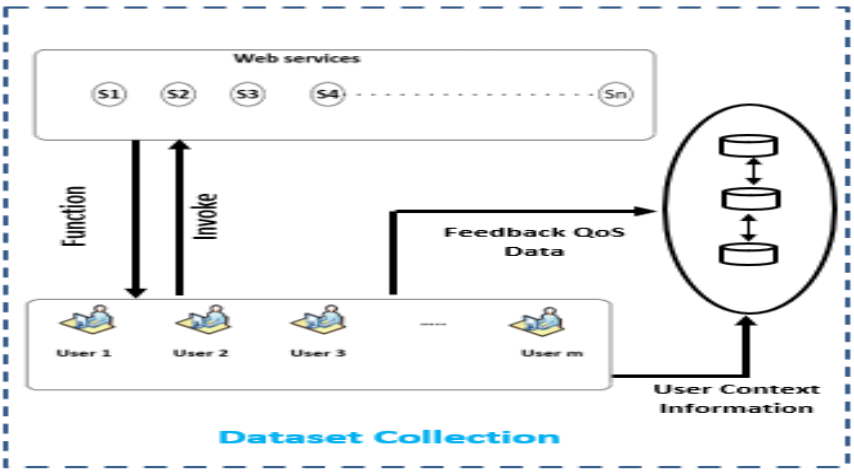


Figure (4.1) QoS Data Collection

By processing the Web service invocation results, two 339 × 5,825 matrices were created for response time attribute and throughput attribute, respectively. Each entry in a matrix represents the response-time value or throughput value observed by a user on a Web service. In each matrix, there exist 1,974,675 QoS records for 5825 web services distributed in 73 countries and observed by 339 users from 30 countries. Figure (4.2) shows the two matrices.

2.337	0.158	0.108	2.283	0.425	0.264	2.294
-1	1.007	0.715	0.616	0.527	0.263	0.581
0.18	0.161	0.207	0.161	0.161	0.268	0.207
0.428	7.026	-1	0.363	0.184	-1	-1
0.101	0.195	0.404	0.678	0.442	0.102	1.025
1.202	8.046	1.993	4.795	4.394	4.7	1.308
5.212	1.244	1.209	1.172	0.89	0.91	3.451
0.904	14.701	1.52	1.44	3.676	1.721	1.528
0.296	1.88	0.869	1.611	0.588	2.975	0.306
-1	-1	-1	-1	-1	-1	-1
0.292	0.301	0.292	0.296	0.294	0.294	0.294
0.056	0.172	5.473	0.828	1.031	1.272	0.95
0.273	0.839	0.282	0.896	0.306	0.306	1.176
0.822	0.16	0.675	3.671	0.659	0.129	0.151
0.648	0.648	0.886	10.669	-1	-1	3.399
0.236	1.455	0.207	2.64	0.358	0.246	0.243
0.282	0.282	0.281	0.281	0.276	0.283	0.281

(a) Response time matrix

62.999	12.765	11.764	7.594	63.829	1.112	0.379	14.388
9.925	7.633	14.981	0.45	0.604	0.334	7.722	0.492
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
0.388	21.276	1.164	20.689	15.306	24.59	0.524	24.793
4.962	0.337	5.555	0.331	5.464	2.244	5.747	0.551
366.013	6.972	363.636	0.343	0.882	87.719	0.523	0.532
790.419	440.944	-1	522.058	827.777	642.857	476.19	-1
0.332	0.526	173.553	22.727	27.027	2.573	13.26	1.664
0.53	25.0	0.536	25.0	25.21	2.518	2.066	0.5
0.515	12.658	0.468	0.716	1.864	0.52	0.65	0.517
21.276	11.764	21.582	0.69	250.0	-1	15.873	1.078
28.169	0.362	0.512	0.546	0.344	21.546	21.438	4.518
49.019	45.871	37.037	40.65	46.296	40.0	49.019	46.296
159.292	20.408	10.309	9.039	966.666	634.146	-1	383.177
6.185	6.835	0.518	15.625	0.528	66.666	-1	15.384
6.42	24.193	24.0	38.961	122.302	121.428	8.347	22.9
-1	-1	-1	-1	-1	-1	806.451	-1
187.5	7.009	30.303	48.387	47.619	10.714	52.631	63.829

(b) Throughput matrix

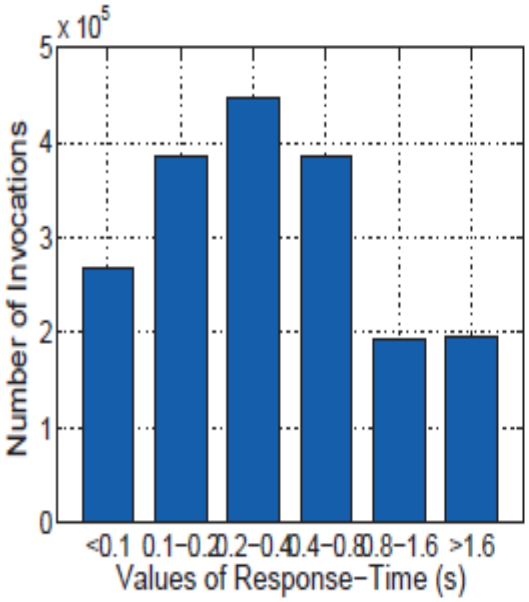
Figure (4.2) QoS Attribute Matrices

WS -Dream dataset contains two other files that represent the user profile and the service profile. The user profile consists of the user's context information and includes seven attributes that describe basic information for users: User ID, IP Address, Country, IP Number, Autonomous System (AS), Latitude, and Longitude. While The service profile consists of the service's context information and includes nine attributes that describe basic information for services: Service ID, WSDL Address, Service Provider, IP Address, Country, IP Number, Autonomous System (AS), Latitude, and Longitude. Figure (4.3) shows the user profile matrix.

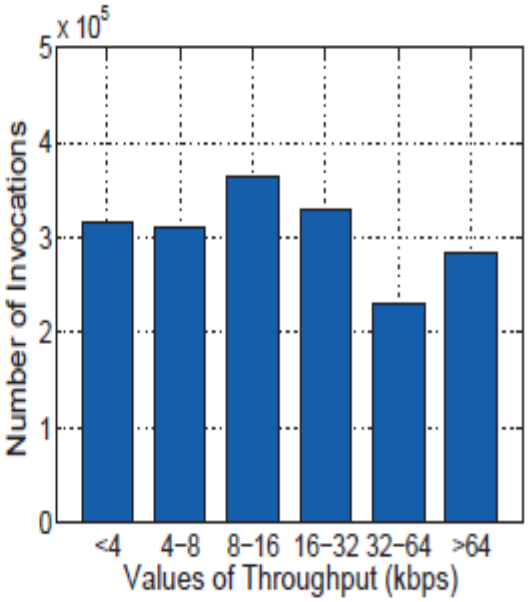
[User ID]	[IP Address]	[Country]	[IP No.]	[AS]	[Latitude]	[Longitude]
67	129.242.19.197	Norway	2180125637	AS224 UNINETT, The Norwegian University & Research Network	69.6667	18.9667
68	129.59.88.180	United States	2168150196	AS7212 Vanderbilt University	36.1473	-86.777
69	129.69.210.96	Germany	2168836704	AS553 Landeshochschulnetz Baden-Wuerttemberg (BelWue)	48.7667	9.1833
70	129.69.210.97	Germany	2168836705	AS553 Landeshochschulnetz Baden-Wuerttemberg (BelWue)	48.7667	9.1833
71	129.74.74.19	United States	2169129491	AS693 University of Notre Dame	41.7007	-86.2501
72	129.74.74.20	United States	2169129492	AS693 University of Notre Dame	41.7007	-86.2501
73	129.93.229.138	United States	2170414474	AS7896 University of Nebraska-Lincoln	40.8	-96.667
74	129.93.229.139	United States	2170414475	AS7896 University of Nebraska-Lincoln	40.8	-96.667
75	129.97.74.12	Canada	2170636812	AS12093 University of Waterloo	43.4529	-80.5281
76	129.97.74.14	Canada	2170636814	AS12093 University of Waterloo	43.4529	-80.5281
77	130.136.254.21	Italy	2190016021	AS137 GARR Italian academic and research network	44.4833	11.3333
78	130.149.49.136	Germany	2190815624	AS680 service G-WiN	52.5167	13.4
79	130.149.49.137	Germany	2190815625	AS680 service G-WiN	52.5167	13.4
80	130.192.86.30	Italy	2193643038	AS137 GARR Italian academic and research network	45.05	7.6667
81	130.206.158.140	Spain	2194579084	AS766 RedIRIS Autonomous System	42.8141	-1.6412
82	130.237.50.124	Sweden	2196583036	AS1653 SUNET Swedish University Network	59.3333	18.05
83	130.75.87.83	Germany	2185975635	AS680 service G-WiN	52.3667	9.7167
84	130.83.166.198	Germany	2186520262	AS8365 Man-da.de GmbH	49.8706	8.6494
85	130.83.166.199	Germany	2186520263	AS8365 Man-da.de GmbH	49.8706	8.6494
86	130.83.166.200	Germany	2186520264	AS8365 Man-da.de GmbH	49.8706	8.6494
87	130.92.70.252	Switzerland	2187085564	AS559 SWITCH, Swiss Education and Research Network	46.9167	7.4667
88	130.92.70.253	Switzerland	2187085565	AS559 SWITCH, Swiss Education and Research Network	46.9167	7.4667
89	131.130.32.153	Austria	2206343321	AS760 University of Vienna, Austria	48.2	16.3667
90	131.130.32.154	Austria	2206343322	AS760 University of Vienna, Austria	48.2	16.3667
91	131.175.17.10	Italy	2209288458	AS137 GARR Italian academic and research network	42.8333	12.8333

Figure (4.3) User Profile Matrix

Figure (4.4) shows the distribution of response time and throughput. Figure (4.4.a) shows that most of the response-time values are smaller than 1.6 seconds. Figure (4.4.b) shows that most throughput values are smaller than 64 kbps.



(a) Distribution of response time



(a) Distribution of throughput

Figure (4.4) QoS Values Distribution

A detailed statistical description for QoS data set of web services is presented in Table (4.1)

Table (4.1) Statistic of Web Service QoS Dataset

Statistics	Values
Num. of Records	1,974,675
Num. of Service Users	339
Num. of Web Services	5,825
Num. of User Countries	30
Num. of Web Service Countries	73
Scale of Response Time	0-20s
Mean of Response-Time	1.43 s
Standard Deviation of Response-Time	31.9 s
Scale of Throughput	0-1000kbps
Mean of Throughput	102.86 kbps
Standard Deviation of Throughput	531.85 kbps

4.3 Preprocessing Step

Initially, the original QoS dataset (response time and throughput) has been converted from text format to matrix format. Preprocessing step was taken place through the following processes:

- Deleting the services that have not been invocated by any user (empty records).
- Normalizing QoS values in order to divide the statistical interval more conveniently.

Table (4.2) illustrates a sample of the response time matrix after being pre-processed.

Table (4.2) Normalizing QoS Value of Response Time Matrix

	S_0	S_1	S_2	S_3	S_4	S_5	...	S_{5819}	S_{5820}	S_{5821}	S_{5822}	S_{5823}	S_{5824}
u_0	0.34	0.04	0.01	0.05	0.05	0.08	...	0.33	0.37	0.40	0.35	0.43	0.70
u_1	0.11	0.04	0.01	0.06	0.06	0.06	...	0.04	0.02	0.02	0.15	0.01	0.02
u_2	0.03	0.07	0.02	0.09	0.09	0.02	...	0.09	-1	-1	0.69	-1	-1
u_3	0.02	0.04	0.01	0.05	0.05	0.05	...	0.02	0.01	0.01	0.01	0.01	0.01
	...	...	...	...	...	...	...	...	...	...	...	...	...
u_{335}	0.02	0.04	0.01	0.05	0.05	0.06	...	0.02	0.01	0.01	0.01	0.01	0.01
u_{336}	0.05	0.06	0.02	0.11	0.08	0.10	...	0.11	0.04	0.05	0.04	0.06	0.04
u_{337}	0.05	0.13	0.03	0.98	0.08	0.08	...	0.08	0.03	0.04	0.03	0.08	0.06
u_{338}	0.05	0.03	0.01	0.05	0.05	0.07	...	0.02	0.01	0.01	0.01	0.01	0.01

4.4 User Reputation-based Calculation

The calculation of user reputation is described as follows:

First, we need to calculate the reliable interval of each service, then the user feedback information can be divided into trusted feedback, untrusted feedback and no feedback, according to the QoS value of the web service invoked by the user and reliable interval. After that, we count the feedback information of each user and express it as feedback vector. Finally, the reputation of users is calculated based on Dirichlet probability distribution.

In order to calculate reliable interval of each service, the QoS values in [0,1] intervals can be evenly divided into 10 intervals of (0,0.1], (0.1, 0.2], ..., (0.8,0.9], and (0.9, 1]. Trusted users always take the major part of all users. Table (4. 3) shows how to use statistical intervals to determine the reliable interval in service 0.

Table (4.3) Statistical Interval Results

Service 0

	0.0-0.09	0.1-0.19	0.2-0.29	0.3-0.39	0.4-0.49	0.5-0.59	0.6-0.69	0.7-0.79	0.8-0.89	0.9-0.1	No Invocation
u_0				0.34							
u_1		0.11									
u_2	0.03										
u_3	0.02										
u_4	0.03										
u_5	0.09										
u_6	0.02										
u_7	0.02										
u_8		0.1									
u_9	0.03										
u_{10}				0.36							
...	...	...	...	...	...	...	...	...	...	...	...
u_{81}											-1
...	...	...	...	...	...	...	...	...	...	...	...
u_{336}	0.05										
u_{337}	0.05										
u_{338}	0.05										

Table (4.3) presents an explanation of the results for QoS values of service 0 based on statistical interval. The first column (first interval) has contained most QoS values of users, and can be considered reliable interval for service 0. Table (4.4) shows whether users of service 0 are in the reliable interval, are not in the reliable interval, or have not invoked the service. The user's QoS value in the reliable interval can be considered as trusted feedback. Otherwise, it is untrusted feedback.

Table (4.4) User Feedback Information

Service 0

	If user in reliable intervale?	
u_0		No
u_1		No
u_2	Yes	
u_3	Yes	
u_4	Yes	
u_5	Yes	
u_6	Yes	
u_7	Yes	
u_8		No
u_9	Yes	
u_{10}		
...	...	...
u_{81}	No Invocation	
...	...	...
u_{336}	Yes	
u_{337}	Yes	
u_{338}	Yes	

After completing the calculation of all feedback information for all services, we count each user’s feedback information, and express the user’s feedback information as a feedback vector.

Table (4.5) shows the feedback vector for all users.

Table (4.5) User Feedback Vector

	User Feedback Vector		
	No Feedback	**Untrusted Feedback**	**Trusted Feedback**
u_0	193	2743	2889
u_1	275	572	4978
u_2	307	2005	3513
u_3	283	395	5147
u_4	267	384	5174
u_5	267	541	5017
u_6	273	440	5112
u_7	260	521	5044
u_8	270	412	5143
u_9	243	482	5100
u_{10}	193	3234	2398
...	...	...	...
u_{81}	193	3339	2293
...	...	...	...
u_{336}	300	1736	3789
u_{337}	304	1405	4116
u_{338}	294	624	4907

Finally, the user's reputation can be calculated from the feedback vector by using the probability expectation of the Dirichlet distribution. Table (4.6) shows that the identification of untrusted users depends on the threshold value (0.5). If the reputation value of the user is less than the threshold value, the user is identified as untrusted.

Table (4.6) Identifying Untrusted User

	User Reputation Value	**Trusted/ Untrusted Depending on the threshold (0.5)**
u_0	0.5129	Trusted
u_1	0.8967	Trusted
u_2	0.6365	Trusted
u_3	0.9285	Trusted
u_4	0.9307	Trusted
u_5	0.9025	Trusted
u_6	0.9205	Trusted
u_7	0.9062	Trusted
u_8	0.9256	Trusted
u_9	0.9135	Trusted
u_{10}	0.4258	Untrusted
...	...	...
u_{81}	0.4071	Untrusted
...	...	...
u_{336}	0.6857	Trusted
u_{337}	0.7454	Trusted
u_{338}	0.8870	Trusted

4.5 Clustering Users

One of the essential aspects that need to be taken into consideration are the users in the same geographical location tend to have similar QoS values for the same service, because they have a similar distance to the same provider and similar infrastructure and user that located in different location observed different QoS values for the same services. Thus, the DBSCAN clustering algorithm was used on the user side to form clusters according to latitude and longitude attributes of users with two input parameters, *minpts* = 2 and *Eps.* = 0.5. After implementing the DBSCAN algorithms, we formed 11 clusters.

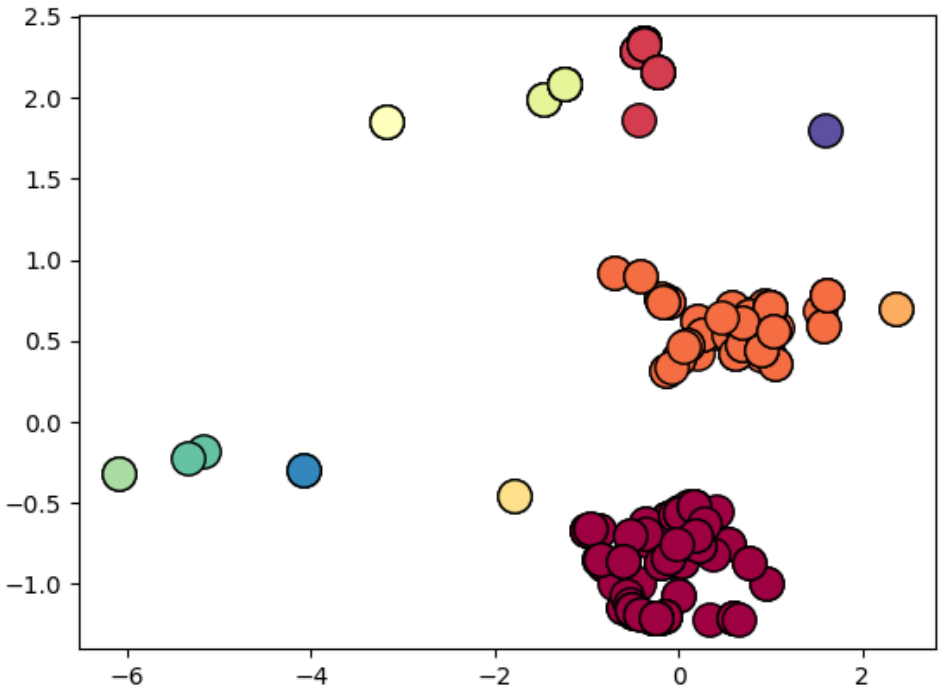


Figure (4.5) Clustering User Geographical Location

4.6 Predicting Unknow QoS values

Predicting Unknown QoS values depends essentially on the historical experience of neighboring users. Therefore, we cluster the users according to their location. After clustering the users, we predict the

unknown QoS values based on the values previously invoked by other neighboring users in the same cluster by computing the cosine similarity between users and then using the values of this similarity as weights.

To predict the invocation value of service S_0 for user u_0 in the table (4.7) that represent cluster 0, the cosine similarity computing between user u_0 and other users that have previously invocation values for S_0. As shown in the table (4.7), users (u1,u2,u4,u6,u8 and u9) have previously invocation for S_0, and the predict the missing value according to Equation (3.13).

Table (7.4) Cluster of QoS values

	Cluster 0									
	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
u_0		0.05	0.02		0.07	0.8		0.09	0.1	0.44
u_1	0.04	0.04		0.06	0.06	0.04	0.05		0.1	
u_2	0.02	0.05	0.01	0.06			0.05			0.35
u_3			0.01		0.05	0.06		0.05	0.07	0.03
u_4	0.04	0.04		0.05	0.05		0.03	0.05		
u_5		0.03	0.01			0.05	0.07		0.07	0.03
u_6	0.02		0.01	0.04	0.04			0.06	0.08	0.03
u_7		0.05			0.07	0.04	0.05	0.08	0.11	
u_8	0.04		0.02	0.07		0.05				0.04
u_9	0.17	0.05		0.06	0.06		0.06	0.09	0.1	0.05
u_{10}		0.05	0.01		0.05	0.15		0.06	0.07	

4.7 Performance Comparison

For demonstrating how accurate the proposed approach is in terms of predictions, the present section will draw a comparison between the results obtained through the proposed approach and those of some well-known QoS prediction approaches, including the following:

1. UPCC (User-based collaborative filtering method using Pearson Correlation Coefficient): The calculation of similarity among users is performed by means of PCC, and the prediction

of missing QoS values takes place according to the user's similarity [92].

2. IPCC (Item-based collaborative filtering method using Pearson Correlation Coefficient): As for the second method, the calculation of similarity between services is performed through PCC, and the prediction of missing QoS values occurs according to the service's similarity [93].
3. UIPCC: (User-Item-based collaborative filtering method using Pearson Correlation coefficient): This represents a hybrid approach whereby UPCC and IPCC are combined, and it utilizes the similarity of both users and services for improving the predicting accuracy [94] .
4. NMF (Nonnegative Matrix Factorization): It employs non-negative matrix factorization onto the user-service matrix for predicting missing QoS values [95].
5. Cloud-Pred (Cloud Prediction): It is a matrix factorization approach that uses both local and global usage information of both users and services when predicting [96].

In reality, the matrices of QoS attributes are very sparse. To evaluate the impact of matrix density on the prediction accuracy, some values are removed randomly from the matrices and the prediction results are compared with the results of other approaches. To exemplify, if the matrix density is 90%, this implies that 10% of the entries in the original matrix are randomly removed, leaving the remaining 90% as test data.

The experimental results of MAE and RMSE for various approaches at different densities are shown in Table (4.2) and Table (4.3), which in turn highlight the result of the proposed approach.

Table (4.2) Response-Time Performance Comparison Using MAE and RMSE Metrics.

Matrix density (%)	Metrics	Response-Time (seconds)					
		IPCC	UPCC	UIPCC	NMF	CloudPred	Proposed Approach
10	MAE	0.5173	0.5655	0.5654	0.6754	0.5306	**0.4710**
	RMSE	1.4874	1.3326	1.3309	1.5354	1.2904	**1.2693**
20	MAE	0.5135	0.5516	0.5053	0.6771	0.4745	**0.3506**
	RMSE	1.3419	1.3114	1.2486	1.5241	1.1973	**1.0942**
80	MAE	0.4785	0.4442	0.3873	0.3740	0.3704	**0.3159**
	RMSE	1.3414	1.1514	1.0785	1.1242	1.0597	**1.0031**
90	MAE	0.4605	0.4331	0.3793	0.3649	0.3638	**0.3031**
	RMSE	1.2761	1.1264	1.0592	1.1121	1.0359	**0.9570**
Average of MAE		0.4924	0.4986	0.4593	0.5228	0.4348	**0.3601**
Average of RMSE		1.3617	1.2304	1.1793	1.3239	1.1458	**1.0809**

Table (4.3) Throughput Performance Comparison Using MAE and RMSE Metrics.

Matrix density (%)	Metrics	Throughput (kbps)					
		IPCC	UPCC	UIPCC	NMF	CloudPred	Proposed Approach
10	MAE	27.019	26.201	22.656	19.770	19.000	**15.644**
	RMSE	63.002	61.965	57.465	57.376	51.823	**50.507**
20	MAE	26.195	21.931	18.123	15.779	15.420	**12.024**
	RMSE	60.399	56.544	50.043	50.140	44.897	**40.476**
80	MAE	25.558	14.549	12.488	12.510	10.788	**7.7303**
	RMSE	57.762	44.373	39.601	39.202	36.850	**34.335**
90	MAE	23.973	13.876	12.066	11.696	10.472	**7.5865**
	RMSE	54.882	42.553	38.076	36.755	35.922	**34.152**
Average of MAE		25.686	19.139	16.333	14.938	13.920	**10.746**
Average of RMSE		59.011	51.358	46.296	45.868	42.373	**39.867**

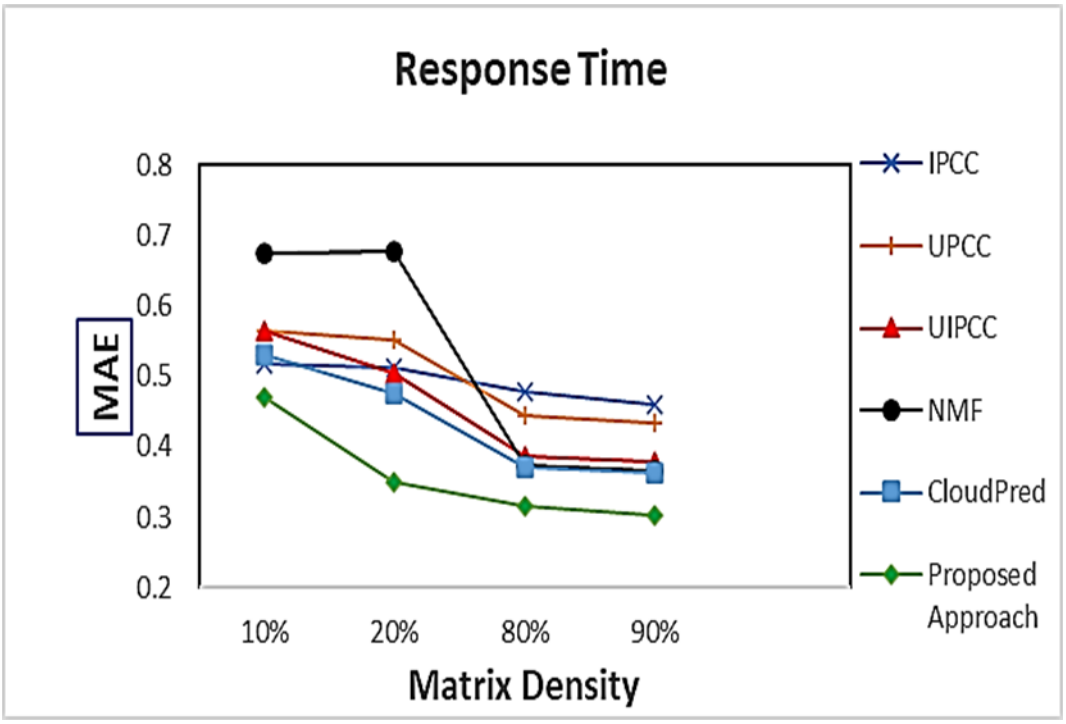
A number of significant aspects are observed in the experimental results:

1. At different matrix densities, it has been observed that that the proposed approach realizes relatively lower RMSE and MAE values than other methods for both response time and throughput attributes.
2. It has been noticed that the values of RMSE and NMA in sparse matrices (e.g., 10%; 20%) are larger than those of dense matrices (e.g., 90%; 80%), which refers to the fact that a sparser matrix provides less information for predicting unknown QoS values.
3. As compared to UIPCC, IPCC, UIPCC, NMF, and Cloud-Pred, the proposed approach obtained more accurate predictions. This can be traced back to the fact that the proposed method makes use of the geographical location of the users to compute the similarity of the neighbors. Thus, users sharing the same location tend to share similar QoS attribute values for the same service. Moreover, the proposed method uses reputation to identify dishonest users to guarantee data reliability by correcting unreliable data using QoS values for trusted users.

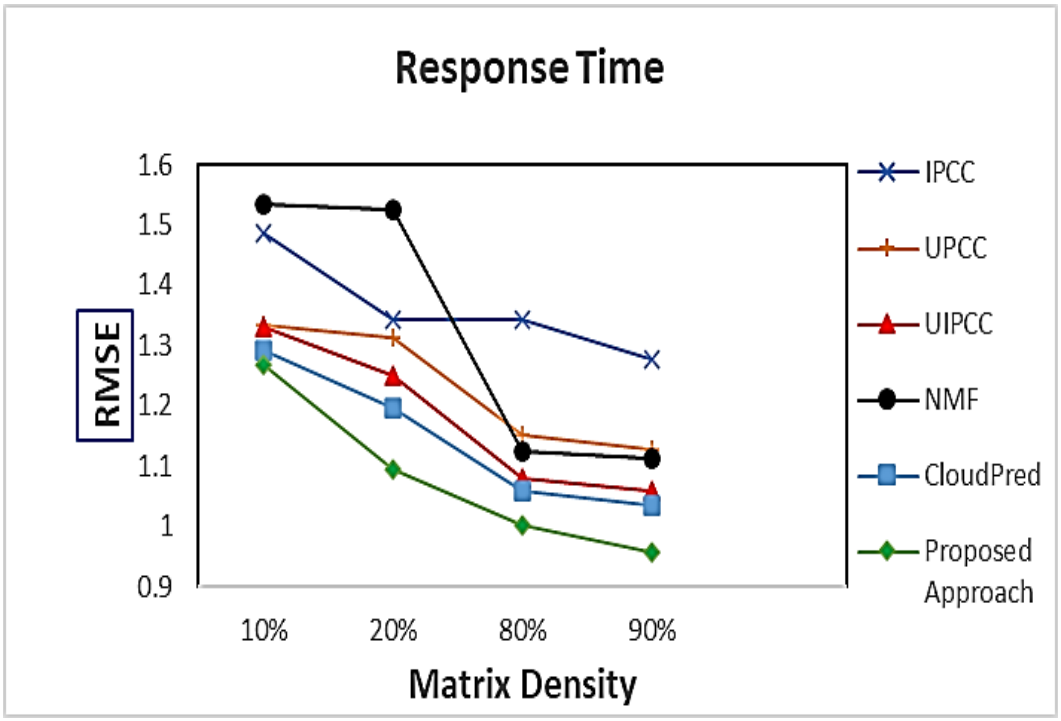
The values of RMSE and MAE of response time are lower than the throughput values, as the scale for the response time is 0-20s, whereas for throughput it is 0-100kbps

4.8 Impact of Matrix Density

The comparison of how accurate the predictions of different approaches at different matrix densities is shown in Figure (4.3) for response time attribute and Figure (4.4) for throughput attribute.

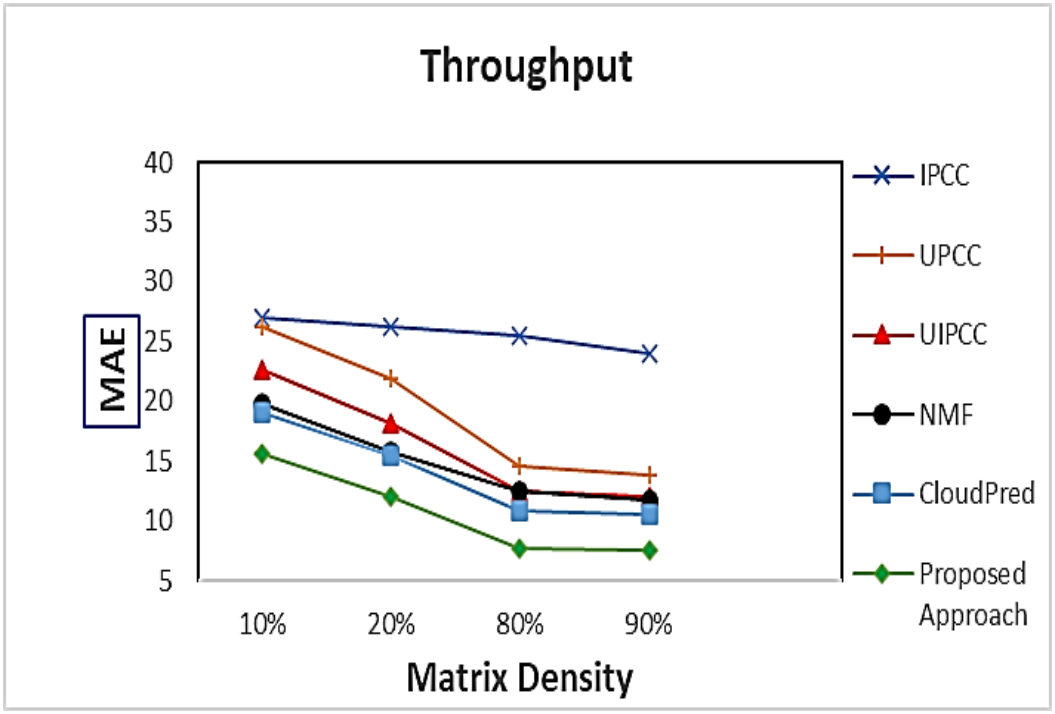


(a) (MAE)

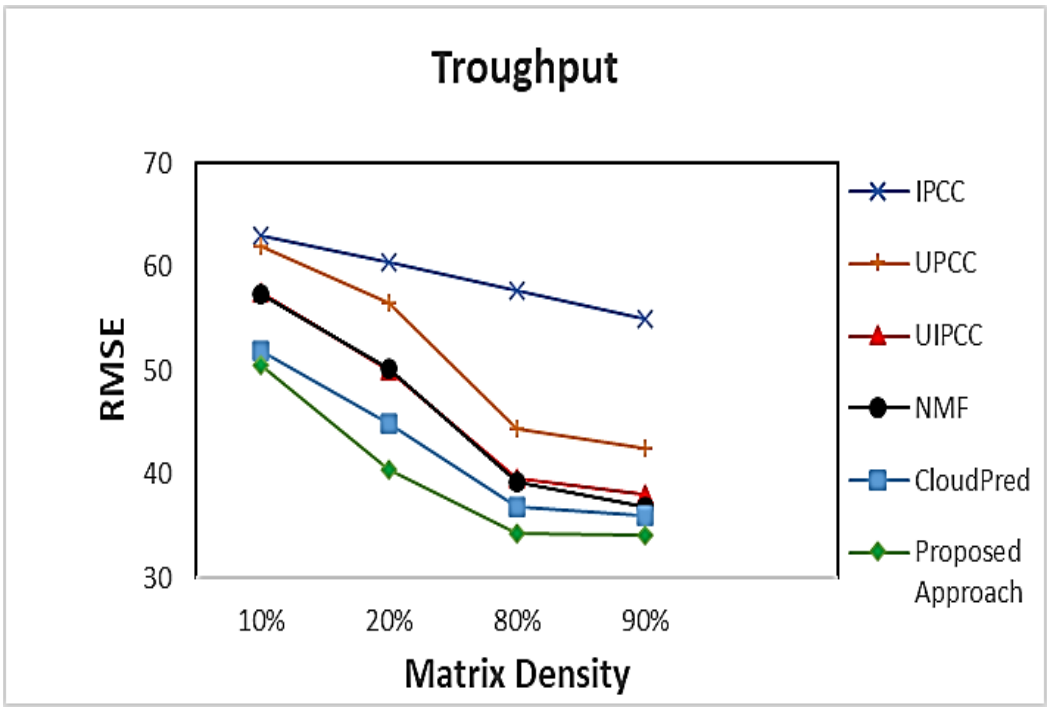


(b) (RMSE)

Figure (4.3) Impact of Response-Time Matrix Density



(a) (MAE)



(b) (RMSE)

Figure (4.4) Impact of Throughput Matrix Density

Figure (4.3) and Figure (4.4) show the accuracy of the prediction improves with the increase in matrix density. This indicates that the increase in matrix density makes more information available for prediction, leading to an increase in the accuracy of the prediction

Chapter Five

Conclusions and Future Works

5.1 Conclusions

Many of the QoS attributes of web services are dynamic and variable due to various factors such as users' reputation and network environment, and their values are different for each user. Therefore, personalized user-side QoS prediction is a necessity and essential to select the best services. This Dissertation,

proposes a new approach for personalized QoS prediction that combines user reputation and user geographic location based on collaborative filtering technique. It could be divided into three steps:

1. The user reputation is calculated using Dirichlet probability distribution for identifying dishonest users and then correct their unreliable data. The purpose of this step is for improving how accurate the QoS predictions are.
2. In order to have an effective similarity computation for users and also to alleviate the data sparsity problem, the users are clustered by their geographic location. As a result, the size of the computations is significantly reduced.
3. Finally, the missing QoS values are predicted in each cluster by weighting the similarity of neighboring users. This proved to be an effective method to improve prediction accuracy. It has been observed that users who share the same location tend to share QoS attribute values that are alike for the same services.
4. The experimental results indicate that the predictions are more accurate than those that are obtained by means of the traditional methods. As for the proposed approach, the missing QoS values are predicted based on the users' geographic location information only to determine the similarity of neighbors

5.2 Future Works

In this Dissertation, build a personalized QoS prediction based on Reputation and Geographic location. Where, the following are suggestions for future works:

1. Developing the system by combining users' information with services information as well as network environment information, where in this case may be enhancing performance with hybrid algorithm.
2. Designing a model for predicting unknown QoS values based on time-aware to improve the accuracy of prediction, since the response time attribute and the throughput attribute are strongly affected by the network environment using machine learning techniques to improve the prediction accuracy.

References

[1] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred : A Time-Aware Personalized QoS Prediction Framework for Web Services," *2011 22nd IEEE Int. Symp. Softw. Reliab. Eng. WSPred*, 2011, doi: 10.1109/ISSRE.2011.17.

[2] Y. Yang, Z. Zheng, X. Niu, and M. Tang, "A Location-Based Factorization Machine Model for Web Service QoS Prediction," *IEEE Trans. Serv. Comput.*, vol. PP, no. c, p. 1, 2018, doi: 10.1109/TSC.2018.2876532.

[3] Z. Chen, L. Shen, F. Li, D. You, and J. P. B. Mapetu, "Web service QoS prediction: when collaborative filtering meets data fluctuating in big-range," *World Wide Web*, vol. 23, no. 3, pp. 1715–1740, 2020, doi: 10.1007/s11280-020-00787-x.

[4] C. Wu, W. Qiu, Z. Zheng, X. Wang, and X. Yang, "QoS Prediction of Web Services based on Two-Phase K-Means Clustering," 2015, doi: 10.1109/ICWS.2015.31.

[5] S. Muthuraman, "Qualitative and Quantitative Review of QOS based Web Services Selection and Composition Techniques STUDY," 2016.

[6] L. Ren and W. Wang, "An SVM-based collaborative filtering approach for Top-N web services recommendation," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 531–543, 2018, doi: 10.1016/j.future.2017.07.027.

[7] R. S. N. and H. N. Nawaf, "QoS Prediction for Web Services Using User Based Classification," *J. Phys. Conf. Ser.*, vol. 1294, no. Issue 4, 2019.

[8] X. Luo, Y. Lv, R. Li, and Y. I. Chen, "Web Service QoS Prediction Based on Adaptive Dynamic Programming Using Fuzzy Neural Networks for Cloud Services," vol. 3, 2015.

[9] T. Chen, R. Bahsoon, and X. Yao, "Online QoS Modeling in the Cloud: A Hybrid and Adaptive Multi-Learners Approach," *2014 IEEE / ACM 7th Int. Conf. Util. Cloud Comput.*, 2014.

[10] M. R. Zhang, Y., Lyu, *QoS Prediction in Cloud and Service Computing Approaches and Applications*. SpringerBriefs in Computer Science, 2017.

[11] L. Kuang *et al.*, "A Personalized QoS Prediction Approach for {CPS} Service Recommendation Based on Reputation and Location-Aware Collaborative Filtering," *Sensors*, vol. 18, no. 5, p. 1556, 2018, doi: 10.3390/s18051556.

[12] S. Ran, "A model for web services discovery with QoS," *ACM SIGecom Exch.*, vol. 4, no. 1, pp. 1–10, 2003, doi: 10.1145/844357.844360.

[13] Y. Zhang, "Modeling and Exploiting QoS Prediction in Cloud and Service Computing," no. September, p. 190, 2013, [Online]. Available: http://search.proquest.com/docview/1547379235/abstract/5C88BAE8A14F49DCPQ/57?accountid=10218%5Cnfiles/2666/Zhang - 2013 - Modeling and Exploiting QoS Prediction in Cloud an.pdf%5Cnfiles/2759/57.html.

[14] S. R. Pat and P. V. P. Mapari, "A Survey On : Web Service Recommendation Using Location-Aware and Personalized Collaborative Filtering," vol. 6, no. 12, pp. 3625–3627, 2016.

[15] S. G. Deng, L. T. Huang, J. Wu, and Z. H. Wu, "Trust-based personalized service recommendation: A network perspective," *J. Comput. Sci. Technol.*, vol. 29, no. 1, pp. 69–80, 2014, doi: 10.1007/s11390-014-1412-2.

[16] S. Deng, L. Huang, and G. Xu, "Social network-based service

recommendation with trust enhancement," *Expert Syst. Appl.*, vol. 41, no. 18, pp. 8075–8084, 2014, doi: 10.1016/j.eswa.2014.07.012.

[17] Q. Xie, S. Zhao, Z. Zheng, J. Zhu, and M. R. Lyu, "Asymmetric correlation regularized matrix factorization for web service recommendation," in *2016 IEEE International Conference on Web Services (ICWS)*, 2016, pp. 204–211.

[18] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *IEEE Trans. ser-vices Comput.*, vol. 4, pp. 140–152, 2011.

[19] M. Tang, Y. Jiang, J. Liu, and X. Liu, "Location-aware collaborative filtering for QoS-based service recommendation," in *2012 IEEE 19th international conference on web services*, 2012, pp. 202–209.

[20] J. Xu, Z. Zheng, and M. R. Lyu, "Web Service Personalized Quality of Service Prediction via Reputation-Based Matrix Factorization," *IEEE Trans. Reliab.*, pp. 1–10, 2015.

[21] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "TAP: A personalized trust-aware QoS prediction approach for web service recommendation," *Knowledge-Based Syst.*, vol. 115, pp. 55–65, 2017.

[22] Z. Chen, L. Shen, and F. Li, "Exploiting Web service geographical neighborhood for collaborative QoS prediction," *Futur. Gener. Comput. Syst.*, vol. 68, pp. 248–259, 2017.

[23] X. Zhu *et al.*, "Similarity-maintaining Privacy Preservation and Location-aware Low-rank Matrix Factorization for QoS Prediction based Web Service Recommendation," *IEEE Trans. Serv. Comput.*, vol. 1374, no. c, pp. 1–14, 2018, doi: 10.1109/TSC.2018.2839741.

[24] L. Chen, F. Xie, Z. Zheng, and Y. Wu, "Predicting Quality of Service via Leveraging Location Information," *Complexity*, vol. 2019, 2019,

doi: 10.1155/2019/4932030.

[25] Z. Chen, Y. Sun, D. You, F. Li, and L. Shen, "An accurate and efficient web service QoS prediction model with wide-range awareness," *Futur. Gener. Comput. Syst.*, vol. 109, pp. 275–292, 2020.

[26] X. Wang, P. He, J. Zhang, and Z. Wang, "QoS Prediction of Web Services Based on Reputation-Aware Network Embedding," *IEEE Access*, vol. 8, pp. 161498–161508, 2020.

[27] M. Hasnain, M. F. Pasha, I. Ghani, M. Imran, M. Y. Alzahrani, and R. Budiarto, "Evaluating trust prediction and confusion matrix measures for web services ranking," *IEEE Access*, vol. 8, pp. 90847–90861, 2020.

[28] M. I. Smahi, F. Hadjila, C. Tibermacine, and A. Benamar, "A deep learning approach for collaborative prediction of Web service QoS," *Serv. Oriented Comput. Appl.*, vol. 15, no. 1, pp. 5–20, 2021.

[29] J. Yin and Y. Xu, "Personalised QoS-based Web service recommendation with service neighbourhood-enhanced matrix factorisation," *Int. J. Web Grid Serv.*, vol. 11, no. 1, pp. 39–56, 2015, doi: 10.1504/IJWGS.2015.067156.

[30] U. Javed, K. Shaukat, I. A. Hameed, F. Iqbal, T. M. Alam, and S. Luo, "A Review of Content-Based and Context-Based Recommendation Systems," *Int. J. Emerg. Technol. Learn.*, vol. 16, no. 3, pp. 274–306, 2021, doi: 10.3991/ijet.v16i03.18851.

[31] W. T. Tsai, X. Sun, and J. Balasooriya, "Service-oriented cloud computing architecture," *ITNG2010 - 7th Int. Conf. Inf. Technol. New Gener.*, pp. 684–689, 2010, doi: 10.1109/ITNG.2010.214.

[32] J. Zela Ruiz and C. M. Rubira, "Quality of Service Conflict during Web Service Monitoring: A Case Study," *Electron. Notes Theor.*

Comput. Sci., vol. 321, pp. 113–127, 2016, doi: 10.1016/j.entcs.2016.02.007.

[33] L. Rodero-Merino *et al.*, "From infrastructure delivery to service management in clouds," *Futur. Gener. Comput. Syst.*, vol. 26, no. 8, pp. 1226–1240, 2010, doi: 10.1016/j.future.2010.02.013.

[34] S. Murugesan and I. Bojanova, *Encyclopedia of cloud computing*. John Wiley & Sons, 2016.

[35] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "NIST cloud computing reference architecture," *Proc. - 2011 IEEE World Congr. Serv. Serv. 2011*, pp. 594–596, 2011, doi: 10.1109/SERVICES.2011.105.

[36] R. Buyya, J. Broberg, and A. Goscinski, *Cloud Computing: Principles and Paradigms*. 2011.

[37] B. Furht and A. Escalante, *Handbook of cloud computing*, vol. 3. Springer, 2010.

[38] K. Manasa Dhara, M. Dharmala, C. Krishan Sharma, K. Manasa, and C. Krishan, "A Survey Paper on Service Oriented Architecture Approach and Modern Web Services," *Surv. Pap. Serv. Oriented Archit. Approach Mod. Web Serv.*, p. 157, 2015, [Online]. Available: http://opus.govst.edu/capstoneshttp://opus.govst.edu/capstones/157.

[39] C. R. Choi and H. Y. Jeong, "A broker-based quality evaluation system for service selection according to the QoS preferences of users," *Inf. Sci. (Ny).*, vol. 277, pp. 553–566, 2014, doi: 10.1016/j.ins.2014.02.141.

[40] Z. Zheng, L. Xiaoli, M. Tang, F. Xie, and M. R. Lyu, "Web Service QoS Prediction via Collaborative Filtering: A Survey," *IEEE Trans. Serv. Comput.*, 2020.

[41] H. F. E. Yamany, M. A. M. Capretz, and D. S. Allison, "Quality of security service for web services with in SOA," *Serv. 2009 - 5th 2009 World Congr. Serv.*, no. PART 1, pp. 653–660, 2009, doi: 10.1109/SERVICES-I.2009.95.

[42] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," *Mob. Networks Appl.*, pp. 1–11, 2019.

[43] K. Kritikos, "Extending OWL for QoS-based web service description and discovery," *CEUR Workshop Proc.*, vol. 169, no. 4, pp. 73–78, 2005.

[44] E. M. Maximilien and M. P. Singh, "A framework and ontology for dynamic web, services selection," *IEEE Internet Comput.*, vol. 8, no. 5, pp. 84–93, 2004, doi: 10.1109/MIC.2004.27.

[45] A. K. Bardsiri and S. M. Hashemi, "QoS Metrics for Cloud Computing Services Evaluation," *Int. J. Intell. Syst. Appl.*, vol. 6, no. 12, pp. 27–33, 2014, doi: 10.5815/ijisa.2014.12.04.

[46] M. Oriol, J. Marco, and X. Franch, "Quality models for web services: A systematic mapping," *Inf. Softw. Technol.*, vol. 56, no. 10, pp. 1167–1182, 2014, doi: 10.1016/j.infsof.2014.03.012.

[47] Q. Tao, H. Y. Chang, C. Q. Gu, and Y. Yi, "A novel prediction approach for trustworthy QoS of web services," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3676–3681, 2012, doi: 10.1016/j.eswa.2011.09.060.

[48] Y. Ma, S. Wang, P. C. K. Hung, C. H. Hsu, Q. Sun, and F. Yang, "A highly accurate prediction algorithm for unknown web service QoS values," *IEEE Transactions on Services Computing*, vol. 9, no. 4. pp. 511–523, 2016, doi: 10.1109/TSC.2015.2407877.

[49] Y. Syu, J. Y. Kuo, and Y. Y. Fanjiang, "Time series forecasting for

dynamic quality of web services: An empirical study," *J. Syst. Softw.*, vol. 134, pp. 279–303, 2017, doi: 10.1016/j.jss.2017.09.011.

[50] Z. Zheng, H. Ma, M. R. Lyu, I. King, and H. Kong, "WSRec : A Collaborative Filtering Based Web Service Recommender System," *2009 IEEE Int. Conf. Web Serv.*, 2009, doi: 10.1109/ICWS.2009.30.

[51] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Trans. Serv. Comput.*, vol. 6, no. 3, pp. 289–299, 2012.

[52] J. Leskovec, A. Rajaraman, and J. D. Ullman, "Mining of Massive Datasets," *Cambridge University Press*. 2020.

[53] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Syst.*, vol. 46, pp. 109–132, 2013, doi: 10.1016/j.knosys.2013.03.012.

[54] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Adv. Artif. Intell.*, vol. 2009, no. Section 3, pp. 1–19, 2009, doi: 10.1155/2009/421425.

[55] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS Prediction for Web Services via Collaborative Filtering," *2007 IEEE Int. Conf. Web Serv.*, no. Icws, 2007.

[56] M. Tang, T. Zhang, J. Liu, and J. Chen, "Cloud service QoS prediction via exploiting collaborative filtering and location-based data smoothing," no. October, pp. 5826–5839, 2015, doi: 10.1002/cpe.

[57] A. Puri, M. Bhonsle, M. E. Student, and G. H. Raisoni, "A Survey of Web Service Recommendation Techniques based on QoS values," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 12. pp. 498–501, 2015, doi: 10.17148/IJARCCE.2015.412142.

[58] P. N. Shendage, “Review on Collaborative Filtering and Web Services Recommendation,” vol. 2, no. 6, pp. 912–916, 2014.

[59] J. Zhu, Y. Kang, Z. Zheng, and M. R. Lyu, “A clustering-based QoS prediction approach for web service recommendation,” *Proc. - 2012 15th IEEE Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput. Work. ISORCW 2012*, pp. 93–98, 2012, doi: 10.1109/ISORCW.2012.27.

[60] M. Fariss, N. El Allali, H. Asaidi, and M. Bellouki, “An Improved Approach for QoS Based Web Services Selection Using Clustering,” *Adv. Sci. Technol. Eng. Syst. J.*, vol. 6, no. 2, pp. 616–621, 2021, doi: 10.25046/aj060270.

[61] J. Liu, M. Tang, Z. Zheng, X. F. Liu, and S. Lyu, “Location-Aware and Personalized Collaborative Filtering for Web Service Recommendation,” *IEEE Trans. Serv. Comput.*, vol. 9, no. 5, pp. 686–699, 2016, doi: 10.1109/TSC.2015.2433251.

[62] S. Kumar, M. K. Pandey, A. Nath, K. Subbiah, and M. K. Singh, “Comparative study on machine learning techniques in predicting the QoS-values for web-services recommendations,” *Int. Conf. Comput. Commun. Autom. ICCCA 2015*, pp. 161–167, 2015, doi: 10.1109/CCAA.2015.7148398.

[63] H. Wu, Z. Zhang, J. Luo, K. Yue, and C. H. Hsu, “Multiple Attributes QoS Prediction via Deep Neural Model with Contexts,” *IEEE Trans. Serv. Comput.*, vol. PP, no. X, p. 1, 2018, doi: 10.1109/TSC.2018.2859986.

[64] G. White, A. Palade, C. Cabrera, and S. Clarke, “Autoencoders for QoS prediction at the Edge,” *2019 IEEE Int. Conf. Pervasive Comput. Commun. PerCom 2019*, pp. 1–9, 2019, doi:

10.1109/PERCOM.2019.8767397.

[65] K. Verbert *et al.*, “Context-aware recommender systems for learning: A survey and future challenges,” *IEEE Trans. Learn. Technol.*, vol. 5, no. 4, pp. 318–335, 2012, doi: 10.1109/TLT.2012.11.

[66] A. R. Deshpande and M. Emmanuel, “Context based recommendation methods : a brief review,” *Int. J. Comput. Appl.*, pp. 14–19, 2016.

[67] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone, “Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems,” *RecSys’09 - Proc. 3rd ACM Conf. Recomm. Syst.*, pp. 265–268, 2009, doi: 10.1145/1639714.1639764.

[68] M. H. Zadeh and M. A. Seyyedi, “Qos Monitoring for Web Services by Time Series Forecasting,” *Comput. Sci. Inf. Tech-nology*, 2010.

[69] W. Qiu, Z. Zheng, X. Wang, X. Yang, and M. R. Lyu, “Reputation-aware QoS value prediction of web services,” in *2013 IEEE International Conference on Services Computing*, 2013, pp. 41–48.

[70] J. Yin, W. Lo, S. Deng, Y. Li, Z. Wu, and N. Xiong, “Colbar: A collaborative location-based regularization framework for QoS prediction,” *Inf. Sci. (Ny).*, vol. 265, pp. 68–84, 2014, doi: 10.1016/j.ins.2013.12.007.

[71] J. Xu, Z. Zheng, and M. R. Lyu, “Web service personalized quality of service prediction via reputation-based matrix factorization,” *IEEE Trans. Reliab.*, vol. 65, no. 1, pp. 28–37, 2015.

[72] S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, and F. Yang, “Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems,” *IEEE Trans. Serv. Comput.*, vol. 8, no. 5, pp. 755–767, 2015, doi: 10.1109/TSC.2014.2320262.

[73] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol, *Recommender Systems Handbook*. 2011.

[74] N. Kumar, Y. S. Sneha, J. Mungara, and S. G. Raghavendra Prasad, "A Survey on Data Mining Methods Available for Recommendation System," *2nd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solut. CSITSS 2017*, pp. 278–283, 2018, doi: 10.1109/CSITSS.2017.8447672.

[75] G. Jain, T. Mahara, and K. N. Tripathi, "A Survey of Similarity Measures for Collaborative Filtering-Based Recommender System," *Adv. Intell. Syst. Comput.*, vol. 1053, no. January, pp. 343–352, 2020, doi: 10.1007/978-981-15-0751-9_32.

[76] J. Liu and Y. Chen, "A personalized clustering-based and reliable trust-aware QoS prediction approach for cloud service recommendation in cloud manufacturing," *Knowledge-Based Syst.*, vol. 174, pp. 43–56, 2019, doi: 10.1016/j.knosys.2019.02.032.

[77] D. Xu and Y. Tian, "A Comprehensive Survey of Clustering Algorithms," *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, 2015, doi: 10.1007/s40745-015-0040-1.

[78] S. Sirohi, N. Kumar, and A. Kumar, "a Detailed Study on Clustering Techniques and Tools for Data," *Int. Res. J. Eng. Technol.*, vol. Volume: 05, no. Issue: 04, p. 6, 2018.

[79] A. Fahim, "Homogeneous Densities Clustering Algorithm," *Int. J. Inf. Technol. Comput. Sci.*, vol. 10, no. 10, pp. 1–10, 2018, doi: 10.5815/ijitcs.2018.10.01.

[80] M. Daszykowski and B. Walczak, "Density-Based Clustering Methods," *Compr. Chemom.*, vol. 2, pp. 635–654, 2009, doi: 10.1016/B978-044452701-1.00067-3.

[81] T. Ali, S. Asghar, and N. A. Sajid, "Critical analysis of DBSCAN variations," *2010 Int. Conf. Inf. Emerg. Technol. ICIET 2010*, 2010, doi: 10.1109/ICIET.2010.5625720.

[82] D. Leung, *Denis Trček: Trust and reputation management systems: an e-business perspective*, vol. 20, no. 1–4. 2018.

[83] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, no. 2, pp. 618–644, 2007, doi: 10.1016/j.dss.2005.05.019.

[84] S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, and F. Yang, "Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems," *IEEE Trans. Serv. Comput.*, vol. 8, no. 5, pp. 755–767, 2015, doi: 10.1109/TSC.2014.2320262.

[85] P. Resnick and R. Zeckhauser, *Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System*, vol. 11. Elsevier Science, 2002.

[86] J. Schneider, G. Kortuem, J. Jager, S. Fickas, and Z. Segall, "Disseminating trust information in wearable communities," *Pers. Ubiquitous Comput.*, vol. 4, no. 4, pp. 245–248, 2000, doi: 10.1007/PL00000012.

[87] J. Lin, "On the dirichlet distribution," *Mater's Rep.*, 2016.

[88] A. Jøsang and J. Haller, "Dirichlet reputation systems," *Proc. - Second Int. Conf. Availability, Reliab. Secur. ARES 2007*, pp. 112–119, 2007, doi: 10.1109/ARES.2007.71.

[89] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed qos evaluation for real-world web services," in *2010 IEEE International Conference on Web Services*, 2010, pp. 83–90.

[90] M. M. Suarez-Alvarez, D. T. Pham, M. Y. Prostov, and Y. I. Prostov,

"Statistical approach to normalization of feature vectors and clustering of mixed datasets," *Proc. R. Soc. A Math. Phys. Eng. Sci.*, vol. 468, no. 2145, pp. 2630–2651, 2012, doi: 10.1098/rspa.2011.0704.

[91] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction forweb services via collaborative filtering," in *Ieee international conference on web services (icws 2007)*, 2007, pp. 439–446.

[92] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," *arXiv Prepr. arXiv1301.7363*, 2013.

[93] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[94] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 39–46.

[95] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[96] Y. Zhang and M. R. Lyu, *QoS Prediction in Cloud and Service Computing: Approaches and Applications*. Springer, 2017.

المستخلص

في السنوات الأخيرة ، أصبحت الحوسبة السحابية أكثر شعبية وقابلة للتطوير. تعتبر خدمة الوب المكون الاساسي لهذه التقنية. ومع ذلك، تلعب جودة الخدمة (QoS) لخدمات الوب دورًا مهمًا وأساسيا في اختيار أفضل الخدمات من عدد كبير من خدمات الويب ذات الوظائف المماثلة. لذلك، أصبح التقييم الفعال من جانب المستخدم لجودة خدمة الويب مجال بحثي مهم.

في هذه الاطروحة، تم إقتراح نهج هجين يجمع بين طريقة قائمة على السياق مع تقنية تصفية تعاونية (CF) للتنبؤ بقيم جودة الخدمة (QoS) لخدمات الويب واختيار أفضل الخدمات للمستخدمين النشطين باستخدام ما يلاحظه مستخدمين آخرين من جودة الخدمة للخدمات. ومع ذلك، في الطريقة المقترحة، تم استخدام اثنين من المعلومات السياقية الهامة: سمعة المستخدمين والموقع الجغرافي للمستخدمين. يتم استخدام السمعة لتحسين دقة التنبؤ، بينما يتم استخدام الموقع الجغرافي للمستخدين لتحسين حساب التشابه بين المستخدمين والتخفيف من مشكلة تبعثر البيانات.

تم تقسيم الأجزاء المقترحة من هذا العمل إلى ثلاث مراحل ، تتضمن المرحلة الأولى حساب درجة السمعة لكل مستخدم في مجموعة البيانات باستخدام توزيع احتمالية ديريشلي (Dirichlet) لتحديد المستخدمين غير الأمناء ثم تصحيح بياناتهم غير الموثوقة باستخدام معدل الفترة الموثوقة لجودة الخدمة في كل خدمة .المرحلة الثانية، يتم تجميع جميع المستخدمين حسب الموقع الجغرافي، حيث يمتلك المستخدمون الموجودون في نفس المجموعة قيم متقاربة من قيم جودة الخدمة .المرحلة الثالثة، مرحلة التنبؤ، وهي عملية التنبؤ بقيم جودة الخدمة غير المعروفة باستخدام نهج التصفية التعاوني المستند إلى التشابه بين المستخدمين في كل مجموعة.

كانت نتائج الاطروحة مرضية جدا مقارنة مع أعمال سابقة للتنبؤ بقيم جودة الخدمة غير المعروفة. أظهرت النتائج التجريبية أن الطريقة المقترحة تحسن بشكل كبير أداء التنبؤ بجودة الخدمة من حيث الكفاءة والدقة. كانت قياسات MAE و RMSE لسمة وقت الاستجابة، 0.3601 و 1.0809 على التوالي ، وقياسي MAE و RMSE لسمة الإنتاجية، 10.746 و 39.867 على التوالي.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل- كلية تكنولوجيا المعلومات
قسم البرمجيات

توقع جودة الخدمة للتوصية بخدمة الوب إستنادا على السمعة والتصفية التعاونية القائمة على عنقدة المواقع

مقدمة الى مجلس كلية تكنولوجيا المعلومات/جامعة بابل كجزء من متطلبات الحصول على درجة دكتوراه فلسفة في تكنولوجيا المعلومات/ برمجيات

مِن قِبَل

مؤيد نجم عبد الله ياسين

بأِشراف

أ.د. وسام سمير بهية

2021 A.D. **1443 A.H.**

Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Department of Software

QoS Prediction for Web Service Recommendation Based on Reputation and Location Clustering-Aware Collaborative Filtering

A Dissertation
Submitted to the Council of the College of Information Technology for Postgraduate Studies of University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information Technology- Software

Muayad Najm Abdullah Yaseen

Supervised by

Prof. Dr. Wesam S. Bhaya

2021 A.D. **1443 A.H.**

بسم الله الرحمن الرحيم

﴿ قَالُوا سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا ۖ إِنَّكَ أَنتَ الْعَلِيمُ الْحَكِيمُ (٣٢) ﴾

صدق الله العلي العظيم
سورة البقرة: الآية (٣٢)

Declaration

I hereby declare that this dissertation, submitted to University of Babylon in partial fulfillment of requirements for the degree of Doctorate of Philosophy in Information Technology-Software has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose sources are appropriately cited in the references.

Signature:

Name: Muayad Najm Abdullah

Date: / / 2021

Supervisor Certification

I certify that this dissertation titled "**QoS Prediction for Web Service Recommendation Based on Reputation and Location Clustering-Aware Collaborative Filtering**" was prepared under my supervision at the Department of Software / College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Doctor of Philosophy in Information Technology- Software.

Signature:

Name: Prof. Dr. Wesam S. Bhaya

Date: / / 2021

The Head of the Department Certification

In view of the available recommendations, I forward the dissertation entitled "**QoS Prediction for Web Service Recommendation Based on Reputation and Location Clustering-Aware Collaborative Filtering**" for debate by the examination committee.

Signature:

Name: Asst. Prof Dr. Ahmed Saleem Abbas

Date: / / 2021

Certification of the Examination Committee

We, the undersigned, certify that (Muayad Najm Abdullah) candidate for the degree of Doctor of Philosophy in Information Technology-Software, has presented his dissertation of the following title (**QoS Prediction for Web Service Recommendation Based on Reputation and Location Clustering-Aware Collaborative Filtering**) as it appears on the title page and front cover of the dissertation that the said dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: (19/10/2021).

Signature:
Name: Dr. Huda Naji Nawaf
Title: Professor
Date: / / 2021
(**Chairman**)

Signature:
Name: Dr. Soukaena Hassan Hashen
Title: Professor
Date: / / 2021
(**Member**)

Signature:
Name: Dr. Hawraa Hassan Abbas
Title: Professor
Date: / / 2021
(**Member**)

Signature:
Name: Dr. Mazin Sameer Ali
Title: Assistant Prof**essor**
Date: / / 2021
(**Member**)

Signature:
Name: Dr. Ali Hadi Hasan
Title: Assistant Professor
Date: / / 2021
(**Member**)

Signature:
Name: Dr. Wesam S. Bhaya
Title: Professor
Date: / / 2021
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Dr. Hussein Atiya Lafta
Title: Professor
Date: / / 2021
(**Dean of Collage of Information Technology**)

Dedication

I dedicate the accomplishment of this dissertation to the my family.

Muayad

Acknowledgments

First of all, I would like to thank ***Allah***, The Creator, for His help and acceptance of my prayers that make the accomplishment of this work more than a dream after all what happened.

I would like to express my deep thanks and gratitude to my supervisor Prof. Dr. Wesam S. Bhaya for his valuable guidance and encouragement during the development of my work.

I would also like to thank the staff members of Information Technology College for their faithful efforts to give us the most scientific topics and endless support in all directions aiming to bring into perfection their scientific followers, and for their unforgettable kind and wise management to our affairs during the research period. I'm very grateful and thankful to the Dean of Information Technology College Prof. Dr. Hussein Atiya Lafta.

Last but not least, words cannot express my gratitude and indebtedness to my family, especially my sympathetic, compassionate and beloved mother, my dear brothers, my sister, my dear wife, and my beloved sons. Words cannot describe their constant love, care, concern, patience, and direction in every aspect of my life throughout the years of my study. I'm forever thankful, grateful, and indebted to them.

Muayad Najm Abdullah

Abstract

In recent years, cloud computing has become more popular and scalable services. The core component of this technology is the web service. However, Quality of Service (QoS) for web services seems to have an essential role when it comes to select the best web services from a large number of web services with similar functionality. Therefore, evaluating the user-side efficiently in terms of quality of web services has become a critical research topic.

In this dissertation, a hybrid approach is proposed that combines a context-based method with a collaborative filtering technique to predict the QoS values of web services and select the best services for active users based on historical observations of other users' services. However, in the proposed approach, two important contextual information were used: the reputation and the geographic location of the users. The reputation is used to improve the accuracy of the prediction, while the geographical location is used to improve the similarity calculation of the users and to mitigate data sparsity problem.

The proposed parts of this work have been divided into three stages. In the first stage, a set of untrusted users is identified through the Dirichlet probability distribution on the basis of the user's reputation, followed by processing the unreliable data provided by dishonest users using the mean of the reliable interval in each service. The second stage, clustering all users by geographic location, of which, the users that located in the same cluster have similar QoS values for the same services. In the third stage, the unknown QoS values are predicted using a collaborative filtering approach based on the similarity between users in each cluster.

The dissertation results were very satisfactory compared to the previous collaborative filtering approaches for predict unknown QoS values. The experimental results show that the proposed approach significantly improves the performance of QoS prediction in terms of efficiency and accuracy. The measures of MAE and RMSE for the response time attribute were 0.3601 and 1.0809, respectively, and the measures of MAE and RMSE for the throughput attribute were 10.746 and 39.867, respectively.

Declaration Associated with this Thesis

Some of the works presented in this dissertation have been published as listed below.

1. The work entitled "Predication of Quality of Service (QoS) in Cloud Services: A Survey" has been published in International Conference of Modern Applications on Information and Communication Technology 2020 (ICMAICT), IOP Publishing. The classification of this publishing is related to the Scopus database.

2. The work entitled "Predicting QoS for Web Service Recommendations Based on Reputation and Location Clustering with Collaborative Filtering" has been published in 7th International Conference on Contemporary Information Technology and Mathematics 2021 (ICCITM), IEEE Publishing. The classification of this publishing is related to the Scopus database.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

Abbreviation	Meaning
AS	Autonomous System
CaaS	Computation as a Service
CF	Collaborative Filtering
Cloud-Pred	Cloud Prediction
DaaS	Data as a Service
DBSCAN	Density Based Spatial Clustering of Applications with Noise
IaaS	Infrastructure as a Service
IoS	Internet of Services
IP	Internet Protocol
IPCC	(Item-based collaborative filtering method using Pearson Correlation coefficient
IT	Information Technology
MAE	Mean Absolute Error
MF	Matrix Factorization
NIST	National Institute of Standards and Technology
NMF	Nonnegative Matrix Factorization

PaaS	Platform as a Service
PCC	Pearson Correlation Coefficient
QoS	Quality of Services
RMSE	Root Mean Squared Error
RS	Recommender System
SaaS	Software as a Service
SOA	Service-oriented architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery, and Integration
UIPCC	User-Item-based collaborative filtering method using Pearson Correlation Coefficient
UPCC	User-based collaborative filtering method using Pearson Correlation Coefficient
WS	Web Service
WSDL	Web Services Description Language

Chapter One

General Introduction

1.1 Introduction

Recently, many technologies such as Software Oriented Architecture (SOA) and Cloud Computing have found more popularity [1] [2]. The core component of these technologies is web service [3]. The services offered on the web expanded rapidly and founded the web services needed by the users, and recommender system have played a significant role [4]. However, the increasing growth of web services has resulted in a large number of web services with equivalent functionality [5]. Therefore, selecting the best web service from a huge number of web services with the same functionality is one of the challenges in SOA [6] .

Quality of service (QoS) is a quality measure that refers to a set of features for evaluating the performance of candidate services. However, QoS of web services plays an important role in selecting the best service [7]. Therefore, QoS becomes an important research topic in cloud computing for service selection [8] [9].

In a cloud environment, there are two different types of attributes which utilized to evaluate Quality of Services (QoS) for cloud computing services, functional (static) and non-functional (dynamic) attributes. Functional (static) attributes refer to the performance of QoS services on the service provider side such as accuracy, cost, and reliability that is usually unchangeable for different users, while non-functional (dynamic) attributes refer to QoS observation for invocation services at the user side such as response time, throughput, which is not identical for different users due to the unpredictable links at network environment and diverse in contexts [10] [11].

QoS attributes have been identified as the main non-functional features of web services and the need for their application when selecting

web services [12]. On the user-side, it is necessary to conduct a personalized Quality of Service evaluation for each user. However, the user will typically invoke a small number of services to obtain their QoS performance [13]. Since the evaluation of services on the user-side requires the use of all services on the user-side, which is practically impossible. Therefore, many approaches have been developed to accurately predict unknown QoS values of services using historical invocation records of different users.

Collaborative filtering (CF) technique has widely been used in commercial recommendation system for predicting user interests. It recommends services based on similarity measures between users and/or services. However, this technique make prediction based on past observation of similar users [14].

In some cases, users may be unfairly prejudiced when providing QoS values for invoking services. Since such users do not reflect the actual value of the QoS attributes. Therefore, in CF -based techniques, the unreliability of QoS data provided by untrusted users leads to inaccurate prediction, so checking the trustworthiness of users is an important issue to reduce the impact of unreliable data on prediction accuracy [15] [16].

This dissertation presents a robust model for building personalized QoS prediction by using past invocation records based on reputation and geographic information as context to improve prediction accuracy.

1.2 Motivation

Web services in cloud computing with equivalent functionality have two types of attributes: static (functional) and dynamic (non-functional). To display the QoS values of web services, they should be

distinguished individually according to their static or dynamic nature. The static attributes (functional) of each service, such as the cost of the services, are the same for all users, for example, if we perform a traditional search, we will observe the same QoS values. However, for all other dynamic attributes (non-functional), the value of each QoS attribute of web service can vary for each user, depending on the environment of the Internet and the location of the user. On the other hand, dynamic (non-functional) attributes are the QoS values observed by different users of services at the user-side. So, QoS values can be unreliable data provided by untrusted users for some reasons.

To address these problems, a personalized CF-based approach is proposed to predict QoS values in web services in light of the reputation and geographic information using the historical QoS values of other users who invoked the service.

1.3 Problem Statement

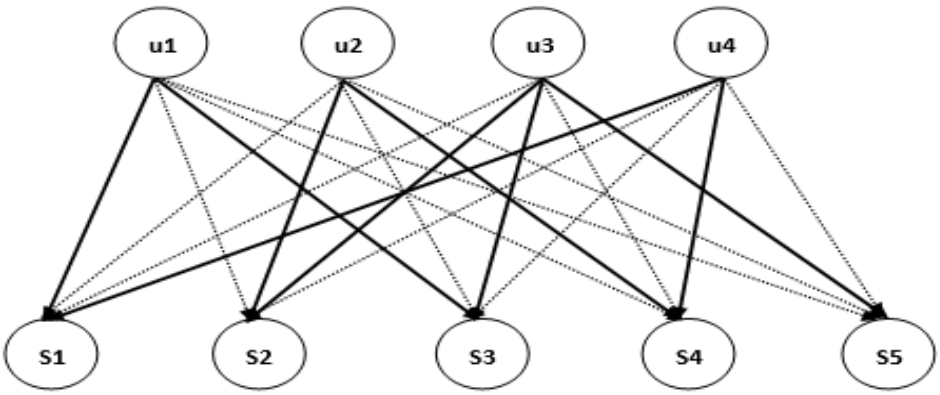
In this section, we will formally state the problem of predicting the QoS values of web services. The QoS values observed by different users of services at the user-side can be represented by a user-service matrix that gives an entry for each user-service pair that represents user's observed QoS value for that service, in which case the problem will be as follows:

1. $U = \{u_1, u_2, \dots, u_m\}$ is the set of users of the web services, where u_i $(1 \leq i \leq m)$ represent a service user.
2. $S = \{s_1, s_2, \dots, s_n\}$ is the set of web services with the same functionality, where s_i $(1 \leq i \leq n)$ represent a web service.

3. Q_{m*n} is the user-service invocation matrix, and $q_{ij} \in R$ shows a value of QoS that the service *j* has for user *i*.

The number of user-service matrices that represent the values of QoS attributes given to web services by users will be the number of QoS attributes. Figure (1.1) shows an example of a user-service matrix for the QoS values of response time and missing value prediction. This matrix shows the response time of each service for each user.

When a user uses a service, the response time or throughput can be calculated through a middleware. However, as the number of web services is very high, each user uses only a small number of services, thus, many entries of these matrices are empty and have no values. Therefore, the prediction problem is to predict the values of the empty entries of this matrix using the existing values of the matrix after correcting the unreliable data.



	S1	S2	S3	S4	S5
u1	0.45		0.3		
u2		0.73		0.32	
u3		0.55	0.3		0.50
u4	0.4			0.46	

User-Services Matrix

	S1	S2	S3	S4	S5
u1	0.45	0.5	0.3	0.38	0.5
u2	0.3	0.73	0.33	0.32	0.55
u3	0.4	0.55	0.3	0.4	0.50
u4	0.4	0.35	0.30	0.46	0.45

Predicated User-Service Matrix

Figure (1.1) Example of Predicting User-service Matrix

1.4 Objectives of the Dissertation

The objectives of this dissertation are to present a model for building a personalized prediction for the QoS of non-functional attributes after detecting untrusted users through user reputation analysis, from which, users and application designers can obtain a suitable service that efficiently satisfies both functional attributes and non-functional attributes of user requirements. The system uses a real-world dataset of service users and QoS values of service invocations by different users to use for the prediction process.

A user's reputation can be determined by analyzing its past observations to identify untrusted users and correct their unreliable data, followed by computing QoS prediction according to users' location information and reliable data.

In this dissertation, a CF -based prediction approach is built by relying on multiple contextual information: User reputation as implicit context and user location information as explicit context to support the selection of optimal service for active user.

1.5 Dissertation Contribution

The main contributions of this Dissertation are:

1. A confidence aware prediction approach is presented that reduces the effect of unreliable QoS values to make a more accurate prediction and then fixed it to avoid data sparsity problem due to the immediate deleting of unreliable values.
2. Since the QoS values of services are affected by the geographical location of different users and the network environment, a hybrid approach is proposed that combines

CF-based technique with context-based method. In this proposed approach, the data volume is reduced by clustering the data so as to decline the computational size and mitigate the impact of data sparsity problem.

1.6 Related Works

Quality of Service (QoS) prediction for web services is widely discussed in the areas of service selection and service composition, and service recommendation in a dynamic environment [8]. In general, QoS prediction methods can be classified into three main categories: CF-based methods, context-based methods, and context-based CF methods that are the combination of Cf-based with context-based approaches. Furthermore, Collaborative filtering can be divided into main types: memory-based algorithms and model-based algorithms. Memory-based algorithms directly use the similarity between neighbors to predict the QoS values, while in model-based algorithms, the QoS values are predicted by the model created from the behavior of other users or services. [17].

Zheng et al. (2011) have presented a hybrid CF approach (UIPCC) User-Item-based Pearson Correlation Coefficient by combining user-based Pearson Correlation Coefficient (PCC) approach and item-based approach to make accurate prediction for QoS values, in which the similarity between users and services used to predict QoS values, and conducted a large-scale distributed QoS assessment of real-world services. This methods suffer from data sparsity and scalability problems [18].

Tang et al. (2012) have proposed a method for predicting QoS values. In this study, geographic information of the user and services is

combined with a memory-based CF method. This method shows that both user location and service location can improve QoS prediction. However, this method indicate that sufficient QoS entries are required for QoS prediction with normal performance [19].

Xu et al. (2015) have proposed a method called RAM based on user reputation and matrix factorization. Once the reputation of users is calculated, users are divided into different grades and then integrated into a matrix factorization (MF) prediction. However, This method may not produce efficient results if inappropriate parameters are selected [20].

Su et al. (2017) have proposed a trust-aware prediction approach called TAP, which uses K-Means algorithm and beta distribution method to compute users' reputation degrees and cluster honest users, and then identify a set of trusted similar users and services for predicting unknown QoS values. This approach mainly solve the problem of untrustworthy users on the accuracy of QoS prediction, but it do not solve the data sparseness problem [21].

Chen et al. (2017) have presented an approach called Geographical Neighborhood based Matrix Factorization (GNMF), this method is a combination of Matrix Factorization and clustering web services using a bottom-up hierarchical neighborhood clustering algorithm in which neighbors are clustered according to the provider, autonomous system (AS), and country, after clustering web services to determine neighborhood information, the similarity between services and their combination with the geographical location of the service provider is used to predict unknown QoS values. However, this study focus only on transferring knowledge from a web service' geographical neighborhood [22].

Zhu et al. (2018) proposed a location-aware method for prediction QoS that consider a user's privacy introduced as a problem of recommender system. It proposed a similarity-maintaining privacy presentation (SPP) strategy to overcome this problem and then used the L1-norm low-rank matrix factorization technique to predict QoS values. This technique used longitude and latitude of services as a location information to present a location-aware method. This method predicts the missing QoS values without considering whether the QoS values are reliable or not [23].

Chen et al. (2019) presented a location-based prediction method (LANFM) that combines the neural networks with a factorization machine to predict the QoS values of the web services. In this method, the location information of the users and web services converted to embedding vector by a neural network, and then a factorization machine used to perform a prediction [24].

Chen et al. (2020) have developed a model called WRAMF to handle the wide-range change problem of QoS data. They could get more accuracy in predicting QoS values by training the model with parallel stochastic gradient descent algorithm [25].

Wang et al. (2020) proposed a trust-aware CF method is presented for predicting QoS values, which first identifies user trust by using a two-phase K-Means algorithm to exclude unreliable users. Then, reputation-based network embedding is used to learn the hidden representations of users. Finally, the user-based CF is used to predict the missing QoS values [26].

Hasnain et al. (2020) proposed an approach for ranking web services was proposed based on a trust score measuring. In this method, a

confusion matrix was used to determine the trust scores of all web services [27] .

Smahi et al. (2021) proposed a deep learning method for predicting QoS values that combines MF technique based on deep autoencoder and clustering to overcome the sparsity problem in CF. In this method, after clustering users and services using a self-organizing map according to their geographical location, a deep autoencoder is trained for each cluster, and finally unknown QoS values are predicted by the trained deep autoencoder with respect to the nearest cluster [28].

Based on previous research work, we can see that most approaches do not consider the both problems of reliable QoS preprocessing and data sparseness in QoS prediction. Therefore, we propose a personalized QoS prediction to solve the two challenges simultaneously based on reputation and location-aware collaborative filtering for web services. The proposed approach can be divided into three steps. In the first step, a set of untrusted users is identified using Dirichlet probability distribution and correct their contributed QoS values with the average value of the reliable QoS interval for each web service. In the second step, all users are clustered by geographic location using the DBSCAN clustering algorithm to improve the prediction accuracy. In the third step, the missing QoS values are predicted using a collaborative filtering approach based on the similarity between users in each cluster.

Abstracts of above previous work for other researchers have shown in Table 1.1.

Table (1.1): Summary of the Mentioned Related Works

No.	Reference	Technique	Dataset
1.	Zheng et al. (2010)	Memory-based	WSdream-dataset2
2.	Tang et al. (2012)	Memory-based & Location-aware	WSdream-dataset2
3.	Xu et al. (2015)	Reputation-aware & Matrix Factorization	WSdream-dataset2
4.	Su et al. (2017)	Trust-aware & Clustering	WSdream-dataset2
5.	Chen et al. (2017)	Matrix Factorization & clustering & Location-aware	WSdream-dataset2
6.	Zhu et al. (2018)	Matrix Factorization & Location-aware	WSdream-dataset2
7.	Chen et al. (2019)	Neural Network & Matrix Factorization & Location-aware	WSdream-dataset2
8.	Chen et al. (2020)	Matrix Factorization & Neural Network	WSdream-dataset2
9.	Wang et al. (2020)	Trust-aware & Clustering	WSdream-dataset2
10.	Hasnain et al. (2020)	Trust-aware	WSdream-dataset2
11.	Smahi et al. (2021)	Deep learning & Matrix Factorization & Clustering	WSdream-dataset2 WSdream-dataset3

1.7 Dissertation Outline

This dissertation contains five chapters. It is organized as follows: -

Chapter Two reviews Cloud Computing, Service Oriented Architecture (SOA), Quality of Services, Recommendation Systems, Clustering algorithm, Reputation and Trust system.

Chapter Three provides the details of the proposed system, which includes the block diagram and the algorithms to implement the system.

Chapter Four discusses the experimental results of the proposed system and compares with other works.

Chapter Five explains the conclusions of the dissertation, and suggestions for future work.

Chapter Two

Theoretical Background

2.1 Introduction

In recent years, most computer software has transformed from native software to Web-based software, and technologies such as software Oriented Architecture (SOA), Internet of services (IoS), and cloud computing have found popularity[10]. However, the services provided on cloud computing grown dramatically, resulting in a multiple of services with similar functionality, making it difficult for consumers to select the best service. Consumers consider Quality of service (QoS) of the web services to select the best service from among them. The prediction of QoS values of the web services and recommendations of the best service based on these values to the consumers is one of the major challenges in the web service area [29]. A suggested way of solving this challenge is through using Recommender Systems (RSs). The initial goal of these systems is to provide the user with service and data recommendations that may interest them based on these demands and tendencies. Recently, RSs have increased in popularity and are used in different topics. This leads to the notion that an RS could possibly be an essential step in helping users identify the service of interest, thus solving the challenge of selecting the best services [30].

2.2 Cloud Computing

Cloud computing is attracting a lot of interest and is being used on a large scale recently [31]. It has evolved into a scalable platform for the use and delivery of services. One of the technical foundations of cloud computing is Service-Oriented Architecture (SOA), where services from different cloud providers are found and integrated over the Internet. SOA is becoming a popular and important framework for building web applications in the era of Web 2.0 [32].

Cloud computing provides shared resources (e.g., infrastructure, platform, and software) as services available over the network [33]. It is used in a variety of applications, including academic, business, government, and many others, not only for cost saving, but also to achieve strategic IT and business goals. However, Cloud services are used where and when needed, and in the amount needed, and only pay for the resources that they actually use [10].

The important characteristics and their description are displayed in Table (2.1) [34].

Table (2.1) Cloud Characteristics.

Cloud Characteristic	Description
On-demand self-service	Computing capabilities (e.g., server time and network storage) can be unilaterally automatically provisioned as needed).
Broad network access	Capabilities are accessible through heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
Resource pooling	Computing resources (e.g., storage, processing, memory, and bandwidth) are pooled to serve multiple consumers, and are dynamically assigned and reassigned according to demand. Customers have no control over the exact location of resources, but may be able to specify location (e.g., country, state, or datacenter).
Rapid elasticity	Capabilities can be elastically provisioned and released commensurate with demand. Available capabilities often appear to be unlimited.
Measured service	Resource use is automatically controlled and optimized through metering capabilities, appropriate to type of service (e.g., storage, processing, bandwidth, and active user accounts).
Multitenancy	Cloud computing is a shared resource that draws on resource pooling as an important feature. It implies use of same resources by multiple consumers, called tenants.

2.2.1 Entities Involved in Cloud-Based Services

The National Institute of Standards and Technology (NIST) has defined a reference architecture for cloud services. The reference architecture defines five unique actors involved in cloud- based services[35]:

1. **Cloud consumer:** A person or entity that uses the services offered by one or more cloud providers.
2. **Cloud provider:** An entity that provides services to cloud consumers under an appropriate deployment model.
3. **Cloud carrier:** One or more entities that provide network services between cloud providers and cloud consumers.
4. **Cloud broker:** An entity that negotiates relationships between cloud providers and cloud consumers.
5. **Cloud auditor:** An entity that can perform an independent verification of the cloud services offered and their usage.

2.2.2 Cloud Services Types

Network resources, computing resources, and applications offered as services to users by major providers. Cloud services can be divided into three main categories based on the type of services that are provided by the cloud provider: "Software as a Service" (SaaS), "Platform as a Service" (PaaS), and "Infrastructure as a Service" (IaaS) [36].

1. **Software as a Service (SaaS)**

Software as a Service (SaaS) is a delivery model for software applications over the Internet, typically on a subscription basis. It alleviates the need for the users to install and run the applications locally

on their computers, relieving them of the responsibility of repairing and maintaining the hardware and software. However, SaaS replaces traditional software usage with a subscription/rental model, reducing the cost of provisioning and managing the user's physical devices. Google Apps is an example SaaS implementation. It offers a number of web applications with similar functionality to traditional office applications.

2. Platform as a Service (PaaS)

Platform as a Service (PaaS) cloud computing models provide an environment for the execution of application services. A service provider hosts the tools, software, and platform for application development and makes them available to application developers so that they can easily program and deploy their application without having to interact directly with the underlying infrastructure. Microsoft Azure and Google App Engine are an examples of PaaS implementation.

3. Infrastructure as a service (IaaS)

In the IaaS model, essential computing infrastructure, such as network equipment, servers, storage, is provided as a service on demand. Customers get these resources as an outsourced service for as long as they need them instead of purchasing them. Amazon EC2 is an example IaaS implementation. In general, IaaS is divided into two categories:

1. **Computation as a Service (CaaS),** where virtual machine-based servers are rented and billed per hour based on virtual machine capacity
2. **Data as a Service (DaaS),** which uses unlimited storage space to store the user's data regardless of its type, billed per Gigabyte for data size and data transfer.

Table (2.2) shows cloud service types and their capability offered to the user.

Table (2.2) Service Type in Cloud Computing

Service Type	Capability Offered to the user
Software as a Service **(SaaS)**	Use of applications that run on the cloud.
Platform as a Service **(PaaS)**	Deployment of applications on the cloud infrastructure; may use supported programming languages, libraries, services, and tools.
Infrastructure as a Service **(IaaS)**	Provisioning of processing, storage, networks, etc.; may deploy and run operating systems, applications, etc.

2.2.3 Cloud Computing Types

Cloud computing can be categorized based on who owns and operates the cloud. Main types of cloud computing are public, private, and hybrid clouds [37].

1. **Public Cloud**, also referred to as external cloud, is the most common type of cloud computing, where services are made available to the general public on a pay-as-you-go basis.
2. **Private Cloud,** also referred to as internal cloud, is created, deployed, and managed behind the enterprise firewall for the exclusive use of the enterprise. The majority of private clouds are used by large corporations or government agencies that prefer a more controlled and secure environment for their data.
3. **Hybrid Cloud** is a combination of the above two types (private cloud and public cloud). A hybrid cloud allows an organization to keep its critical data and applications behind

its own firewall while provisioning less critical ones in the public cloud.

Figure (2.1) shows the main types of cloud computing.

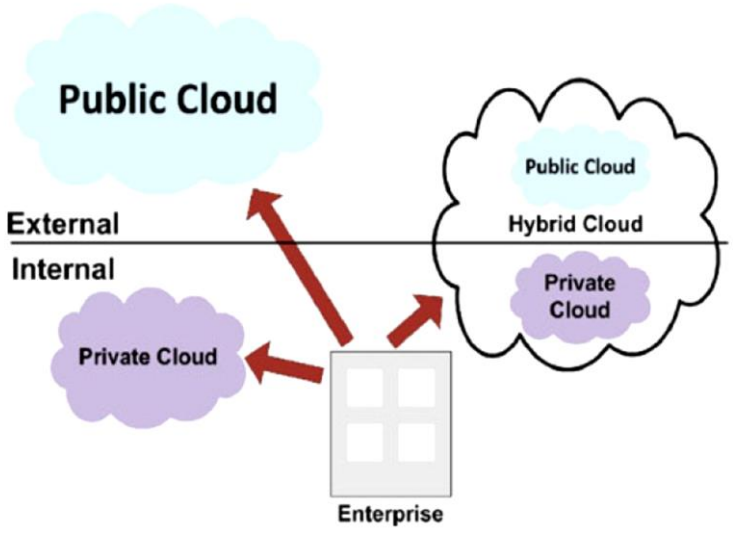


Figure (2.1) Cloud Computing Types

2.3 Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) and cloud computing are related to each other. SOA is an architectural pattern that defines a way to make computing components reusable and interoperable via service interfaces. This interface is a service contract between the service provider and the service user, while cloud computing is a set of enabling technologies that services a bigger, more flexible platform for enterprises to build their SOA solutions. In other words, SOA and cloud computing coexist, complement and support each other [31].

In recent years, web services technology has become the most popular and widely used technology for building SOA applications. Web services are based on a number of protocols and standards such as SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description, Discovery, and

Integration) [38]. Web services are distributed components that are self-contained, discoverable, reusable, composable, and have a transparent location. Due to its popularity, an increasing number of functionally similar web services can be found on the cloud computing, leaving the user to ask the question, "Which are the better services?" or "Which of them fits my needs better?". Users have the difficult task of selecting a suitable service for their requirements [39]. Quality of Service (QoS) has proven to be the most appropriate criterion to distinguish non-functional features between equivalent web services [40].

In SOA, there are three main roles: the service provider, the service user, and the service broker, as shown in Figure (2.2). The service provider defines a service in a WSDL file. This is published to the service broker using UDDI so that the service is discoverable by the service user. The service user requests the service from the service broker. The service broker makes the WSDL file available, and the service user consumes the service directly using SOAP [41].

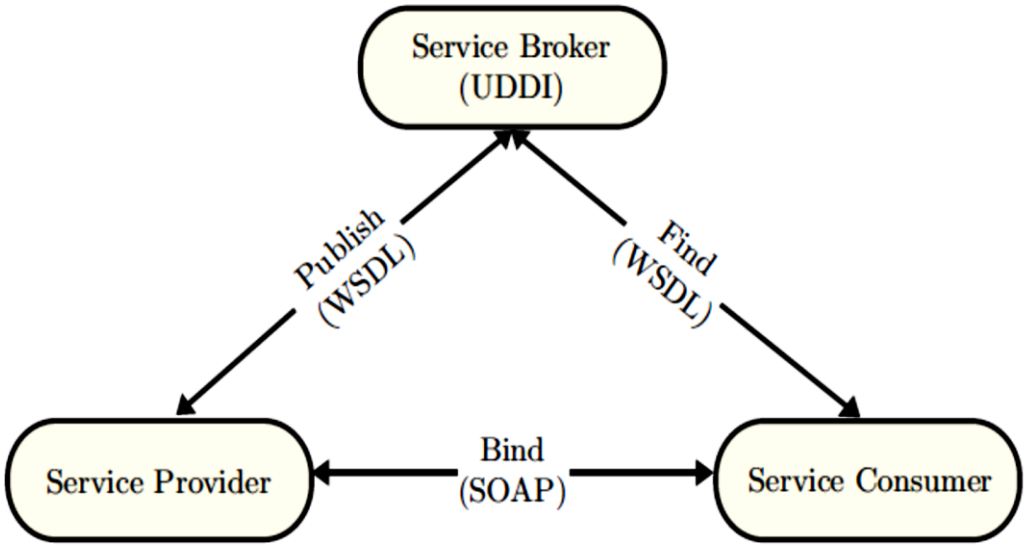


Figure (2.2) SOA Structure

2.4 Quality of Services (QoS)

Quality-of-Service (QoS) is widely employed to represent the non-functional performance of services and has been considered as the key factor to differentiate the qualities of service candidates. It becomes important to evaluate the QoS performance of web services. Therefore, determining the QoS values of web services is a necessity, and most of the QoS attributes have dynamic characteristics, and the values of them are different for every user, and variable in time for a specific user [42].

For the first time in 2003, Ran introduce a service discovery method that considered the non-functional attributes of the Web services and introduced QoS as an important challenge in the Web service domain. Ran defines the quality of services as a set of non-functional attributes that may impact the quality of services offered by a Web service [12]. In another definition, the QoS of Web service defined as "a set of non-functional attributes of the entities used in the path from the WS to the client that bear on the WS' ability to satisfied stated or implied needs in an end-to-end fashion" [43].

Given the importance of QoS in Web services, two issues must be identified about these features:

(1) What are the QoS attributes of the services?

(2) How should these values be determined?

To specify the QoS features, various models are provided for the quality of Web services, and in different researches, metrics defined for Quality-of-Service evaluation [12], [44], [45]. Oriol et al. have systematically compared these models and recognized the QoS attributes of Web service [46]. The most important QoS attributes of Web services

and their definition are displayed in Table (2.٣). Tao divides the QoS attributes into two categories of objective and subjective attributes [47].

Table (2.3) QoS Attributes Definition of Web Services

QoS Attribute	Definition
Accuracy	Error rate produced by the service.
Accessibility	Degree the service is capable of serving a Web service request.
Capacity	Limit of concurrent requests for guaranteed performance.
Response Time	Time to complete a Web service request (from a client perspective).
Throughput	Number of Web service requests served at a given time period.
Availability	The probability that the service can respond to the consumer requests.
MTTR	Meantime to repair.
Interoperability	The ease with which a consumer application or agent interoperates with a service.
Robustness	The degree to which a service can function correctly in the presence of invalid incomplete or conflicting inputs.
Authentication	A measure of how the service authentication principals who can access services and data.
Confidentiality	A measure of how the service threat the data, so that only authorized principals can access or modify the data.
Cost	Cost involved in requesting a service.
Reputation	Measure the user rating to the service.

subjective and rest are objective. Subjective characteristics are known as attributes obtained from the user's subjective evaluation of an items and depend on the morale, interests, and other subjective factors of the user. It considers objective features like the ones not explicitly related to the user's interests. According to this definition, the main QoS attributes of services are in the second category as all the qualities of the services depend somehow on its implementation and the quality of its

construction. For example, reputation, as a subjective parameter, depends on the user' satisfaction with the service provided, and the satisfaction refers to parameters such as availability of the service, response time etc. which are also objective [48]. In another categorization, these attributes are divided into static and dynamic categories. Static QoS attributes are attributes whose value do not change, such as cost, and dynamic QoS attributes are the ones whose values depend on other parameters and can change [49]. Thus, one can say that most of QoS attributes are dynamic and objective.

In response to the second question, how should the values of QoS attributes be determined? Should be said, these features cannot be recorded as other functional features with the service provider's notice in the registry. Because (1) some service providers may announce some unrealistic values for these features. (2) Dynamic QoS features are dependent on the provider, the user, and the network. For instance, the response time of the service depends on the user's location, the time of the service invocation, and the network infrastructure [49]. Thus, the evaluation of services by users can obtain more accurate values for these features [50], [18] on the other hand, evaluating web services from the user's side, which is virtually impossible and has following problems [18], [51]:

1. Using all services needs a lot of time, and some services need a fee that making the process costly.
2. As the number of Web services is high and many Web services are added to these services, some of them may not be evaluated and practically not all services will be deployed.
3. Users are not well-informed enough to be able to measure all of these features.

Considering the above, there should be another way to determine QoS values to be both accurate and feasible. For the first time, Shao uses a collaborative filtering prediction method to determine the values of QoS attributes of Web services [28]. After that, many researchers have also worked on this topic. Thuse, predicting QoS values of services is one of the main approaches for determining the QoS values of Web services.

2.5 Recommendation Systems

There are a wide range of web applications that involve predicting options for users. These applications are called recommender systems (RS). In a recommendation system application, there are two classes of entities, which we shall refer to as users and items. Users have preferences or ratings for certain items, and these preferences or rating must be teased out of the data. The data itself is represented as a utility matrix, giving for each user-item pair [52]. However, Recommender systems collect information about users and items, which can be explicit, such as user ratings, or implicit, such as monitoring the behavior of users [53].

Recommender systems use a number of different techniques. We can divide these systems into three categories [52]:

1. Collaborative filtering technique recommend items based on similarity measures between users and/or items.
2. Context-based technique systems examine properties of the items recommended.
3. A hybrid approach that combines collaborative filtering and context-based techniques.

2.5.1 Collaborative Filtering Technique

One of the most important approaches for predicting user interests in commercial system is collaborative filtering. In these approaches, we focus on the similarity between users or items. The basic idea of CF is that if two users X and Y rate items similarly or exhibit the same behavior, their behavior or rating is similar to another item [54]. This method is widely used in commercial system such Amazon eBay, etc. [18]. For the first time, Shao et al. used the CF method to predict Web service QoS values. They, using similarity between users, predict QoS values of services for a particular user using QoS values of the same service for similar users [55].

In general, CF-based method can be classified into two main categories: memory-based Methods and model-based methods [56].

1. Memory-based Methods

In these algorithms, using a relatively simple statistical equation, QoS values for a user are calculated using the values of similar users or similar services that we know as neighbors. Memory-based or neighborhood-based approaches have two main steps for prediction of QoS values. The first step is similarity computation and the second is predicting the unknown QoS values by the value of similar user or service. One of the main steps in memory-based approaches is to determine the similarity between users or service. The most important similarity measurement method is Pearson Correlation Coefficient (PCC). This criterion has good performance when vectors are continues [57].

After calculated the similarity between users or the similarity between Web services, the next step is to predict QoS values of the services for their users. Usually, at this step, top-k users or services that

are most similar to the active users or serves are selected. Memory-based approaches include user-based, item-based and hybrid algorithms [58] .

A. User-based CF method:

In user-based method, by finding similar users to the active user we can predict the QoS values of services for the active user.

B. Service-based CF method:

In this method, the similarities between different items are calculated and then user rating to an item will be predicted by the rating values of the similar items.

C. Hybrid CF method:

In this method, both the similarities between services and the similarities between users are computed. Then, a hybrid algorithm uses these similarities to predict QoS values. These algorithms have a better result than previous algorithms.

Memory-based algorithms have acceptable accuracy when the user-service matrix is nearly complete, but these methods suffer from some problems [57]:

i. **Data sparsity:** given the high number of web services, users have only used very few numbers of these services, so the predicted QoS values do not have the required accuracy.

ii. **Cold-start:** another problem with the collaborative filtering methods is when a new service or new user is introduced to the system. Therefore, it is not possible to predict QoS values.

iii. **Scalability:** if the number of services and users is very high, then the cost of calculation of the neighbors of services will be

high. Therefore, the scalability problem will be one of the problems of this approach.

iv. **Trust:** In the Cf methods, prediction is made using QoS values provided by other users. Thus, the accuracy of the predicted values will strongly depend on the values contributed by other users. There are several reasons why the values contributed by other users are not correct, such as malicious attacks or comments by competitors, or unrealistic positive comments from the service provider. Hence, distrusted in QoS values contributed by users can be one of the major challenges of CF-based methods.

2. Model-based Methods

These algorithms are presented to solve problems with memory-based methods. In these methods, using a user-service invocation dataset, a pattern is designed, and then by learning from these training techniques such clustering, matrix factorization, time series, and machine learning techniques are among these algorithms [10].

A. Clustering Algorithms

There are many algorithms and criteria for clustering data or object. A group of model-based CF algorithms is methods that cluster users or services. These methods reduce data volumes by clustering the data, increasing scalability, and reducing the volume of computation. Additionally, the cold start problem is also somewhat solved [59], [60].

B. Matrix factorization Algorithms

Matrix Factorization (MF) is the most important model-based approach used for QoS prediction and known as a successful method in

prediction and recommender system. In this method, the user-service matrix $Q = \{q_{ij}\} \in R^{m*n}$ is decomposed to the user feature matrix $U \in R^{d*m}$and the service feature matrix $U \in R^{d*m}$, so the m is the number of users, n is the number of services and d is the number of latent factors. the U and S matrices can be obtained by learning process. The main advantage of this method is that it can somewhat solve the problem of data sparsity and scalability [61]. Despite the advantages, however, these methods have problems, the most important of which are the following:

i. If a user or a new service is added, the model must be recreated and trained. Therefore, the overhead of updating these methods is high.

ii. In most of these methods, there are some parameters that need to be set properly and their values have a significant impact on the prediction accuracy.

iii. These methods do not have the ability to explain the reason for recommending a service, which is a relatively important feature for recommender systems.

C. Machine Learning Techniques

Machine learning techniques are another category of model-based methods that have been used for QoS prediction [62]. In recent researches, machine learning techniques are used for prediction, given that these techniques could model the complex problems with many features well. The most important of these techniques used in predicting QoS values are:

1. Deep neural networks are the most sophisticated approach that can be used to overcome the limitations of matrix factorization. It allows modeling nonlinear interactions in

the data and finding hidden patterns in the data that otherwise could not be detected [63].

2. Some studies used combining fuzzy logic and neural networks to predict unknown QoS values [8].
3. Autoencoder is another prediction method that predicts the QoS values by using a stacked autoencoder, which is a type of neural network [64].

D. Time Series Approaches

Due to the dynamic status of the network and its variable traffic, some QoS attributes such as response time, availability or throughput at different times have different values. In such cases where the value of a property is time-dependent, time series can be a good tool for predicting values [49].

2.5.2 Context-based Technique

Another most commonly used in recommendation system is a context-based technique [52]. The context-based technique makes recommendations to the user based on the properties of items, relevant user information, and the interaction between them [65]. The main steps of context-based based filtering are [66]:

1. Extract the item attributes to create item profile for all items.

2. Create the user profile for each active user.

3. Compare the item profile with user profile.

4. Recommend the items that most closely match the user profile and are not seen by the user.

We can divide context information into three categories:

1. **Computing context:** such as routing information, network properties, hardware and software used.
2. **User context and Item context:** such as geographic location, profile information,
3. **Time context:** such as the time and date of service usage.

Contextual information can be obtained in two ways [67]:

1. Explicit acquisition through directly collected data using registration or rating modules.
2. Implicit acquisition through the analysis of data from service properties and user interactions.

In general, context-based methods can be divided into two main categories: pre-filtering method and post-filtering method [66].

A. **Pre-filtering method:** This method filters the dataset before applying recommendation algorithms.

B. **Post-filtering:** Here the whole dataset is directly used by recommendation algorithms.

2.5.3 Context-aware CF-based Methods

There have been a number of prediction algorithms for web services QoS that combine memory-based or model-based CF techniques with a context-based approach and provide a hybrid approach. In these methods, context information is used by the user or service which can contribute to improving prediction accuracy. Two important contexts employed in these methods are the service or user geographical data and the service invocation time. According to CF methods, the prediction is based on the contributed data from users, so the trustworthiness of this data is an important problem, and for some reason, some of these data may be unreliable. Thus, location, time and trust are three important

contextual factors in determining QoS values of services and increasing the accuracy of prediction [19], [68], [69] .

A. Location-Aware Methods

QoS values such as (response time, availability and throughput) highly dependent on the performance of underlying networks [61]. So, the users that located in the same location have similar QoS attributes values for the same service, because they have a similar distance to the service provider and similar infrastructure and users that located in different location observed different QoS values for the same services [70]. Thus, a category of algorithms, by adding the geographic location of the service or user, or both, in a CF-based manner, try to improve the accuracy of the prediction of QoS values.

B. Trust-Aware Methods

In CF-based prediction methods, new QoS values will be predicted based on the values published by other users. Thus, the accuracy and precision of these values will depend on the accuracy and precision of the values published by the users [71]. Thuse, one of the problems with CF-based methods is invalid data. For avoiding this problem, service or user reputation is usually considered as a parameter. Reputation is a communication parameter derived from the experiences of using a service by the user, reflecting features such as reliability, credits, and integrity of services or user [72]. Moreover, a user's trust in another user or web service can also be used as a criterion to ensure the accuracy of the published data.

2.6 Similarity Measures

Similarity measures are also called similarity metrics. They are methods for calculating scores that express how similar users or items are

to each other. These scores can then be used as the basis for user-based or item-based recommendations. Similarity values can be numeric in the range [-1,1], where the positive value indicates the similarity and the negative difference. Depending on the context of use, similarity metrics can also be referred to as correlation metrics or distance metrics [52].

There are several methods for measuring the similarity between vectors [73] [74] [75]:

1. Jaccard Similarity

The Jaccard similarity of sets is the ratio of the size of the intersection of the sets to the size of the union. This similarity measure is suitable for many applications. For example, if the utility matrix reflects only purchases, this measure would be a good choice. However, when the utility matrices are more detailed ratings, the Jaccard distance loses important information. The equation used in this method of calculating similarity between two users is as follows:

$$\boldsymbol{sim(u_a, u_b) = \frac{I_{u_a} \cap I_{u_b}}{I_{u_a} \cup I_{u_b}}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

Where I_{u_a} and I_{u_b} represent rate values of user a and user b for items, respectively.

2. Euclidean Distances

One of the most important measures of similarity is the Euclidean distance. An *n*-dimensional Euclidean space is a space in which points are vectors of *n* real numbers. The conventional distance measure in this space, called the *L_2-norm*, is defined:

$$\boldsymbol{d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

The generalized form of Euclidean distance is termed as Minkowski Distance. For any constant *r*, we can define the L_r*-norm* to be the distance measured defined by:

$$\boldsymbol{d(x,y) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r}} \quad \text{..... (2.3)}$$

The case *r=2* is the usual L_2*- norm*.

Another common distance measure is the L_1*-norm* or Manhattan distance and is defined as the following equation:

$$\boldsymbol{d(x,y) = \sum_{i=1}^{n} |x_i - y_i|} \quad \text{..... (2.4)}$$

After calculating the distance, similarity is calculated as follows:

$$\boldsymbol{sim(x,y) = \frac{1}{1+d(x,y)}} \quad \text{..... (2.5)}$$

3. Pearson Correlation Coefficient (PCC)

The similarity between any two vectors can also be calculated using Pearson Correlation. It is used to measure the extent to which two variables linearly relate with each other. This criterion has good performance when vectors are continuous. The equation used in this method to compute the similarity between users is as follows:

$$\boldsymbol{sim(u,v) = \frac{\sum_{i \in I}(r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v})}{\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r_u})^2} \sqrt{\sum_{i \in I}(r_{v,i} - \overline{r_v})^2}}} \quad \text{..... (2.6)}$$

Where *u* and *v* are users of the services, the set $I = I_u \cap I_v$ represents the services that both *u* and *v* users have used and have values in the invocation matrix. $r_{u,i}$ is the rate is given by user *u* to the service *i*, and $\overline{r_u}$ is the mean of user's rating u to the various services.

For service similarity computing, PCC used the following equation:

$$sim(i,j) = \frac{\sum_{u \in U}(r_{u,i}-\bar{r}_i)\,(r_{u,j}-\bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i}-\bar{r}_i)^2}\,\sqrt{\sum_{u \in U}(r_{u,j}-\bar{r}_j)^2}} \quad \ldots\ldots (2.7)$$

Where the set $U = U_i \cap U_j$ shows the users who used both service *i* and service *j* in the past and $\bar{r}_i$ is the average of the rate given to service *i*.

4. Cosine Similarity Measure

Measuring cosine similarity is more about the orientation of the two points in space than their exact distance from each other. Therefore, the cosine similarity between the two points is simply the cosine of that angle. If two points were *90* degrees apart, their cosine similarity would be zero because *cos(90)=0*, and if two points were *0* degrees apart, that is, if they were on the same line, their cosine similarity would be *1* because *cos(0)=1*.

Given two vectors *x* and *y*, the cosine of the angle between them is the dot product of the vectors and divided by the lengths of the vectors and defined as the following:

$$sim(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.8)$$

2.7 Clustering Algorithms

Clustering refers to a process where the same data is placed in cluster and data is not similar to the other cluster. Therefore, clustering algorithms look for similarities or dissimilarities among data points. Clustering is an unsupervised learning method so there is no label associated with data points. The algorithm tries to find the underlying structure of the data [54].

The clustering algorithm is used in many QoS prediction methods and service recommendation systems that combine CF -based approaches by clustering users and services according to their geographical location. These algorithms achieved better accuracy in prediction by defining the user and the services region concepts. Also, they could extend scalability by reducing the time complexity of prediction [76].

As with all clustering methods for predicting QoS values of web services, there are two main features:

1. Clustering is only used for data analysis before predicting the original method, so it is always used in combination.
2. The clustering criterion in almost all methods is the geographical similarity of the users or services.

As a result, Clustering with improves the scalability of predictive methods by reducing the data space of services and users. Moreover, clustering with similar services or similar users can reduce the impact of data sparsity [76].

There are several types of clustering algorithms [77] [78] [79]:

1. Hierarchical algorithms create hierarchical analytics for a dataset. In these algorithms, each data point is initially treated as a single cluster. At each iteration, the hierarchical pattern could be subclassified as either a conglomerate or a divider, depending on the shape of the destruction.

2. Partitioning algorithms create k-partitions for a given dataset, as each partition represents a particular group. k-means and k-medoids are the two best-known examples of this type of clustering algorithm. In k-means, each group represents an

average of the centroid or data points within the cluster, while in k-medoids, the groups are represented by the medoids of the cluster, which are the data points located near the cluster center.

3. Density-based algorithms create clusters according to density, as they could identify arbitrarily shaped clusters. The basic idea of these algorithms is that the data which are in the high-density region of the data space are assigned to the same cluster. However, the given clusters are developed further whenever the number of objects or data points within the neighboring area exceed some threshold. DBSCAN algorithm is a well-known example of such an algorithm that forms clusters based on density connectivity analysis.

4. Grid-based algorithms, instead of applying clustering directly to data objects, but rather place data objects into grid cells by partitioning each dimension into a finite number of cells of equal length. Then, some statistical information about the objects in each cell is computed.

5. Model-based algorithms, the basic idea in these algorithms is to select a particular model for each cluster and find the best fit for that model. There are mainly two types of model-based clustering algorithms, one is based on statistical learning method and the other is based on neural network learning method.

Figure (2.3) shows clustering algorithms types [79].

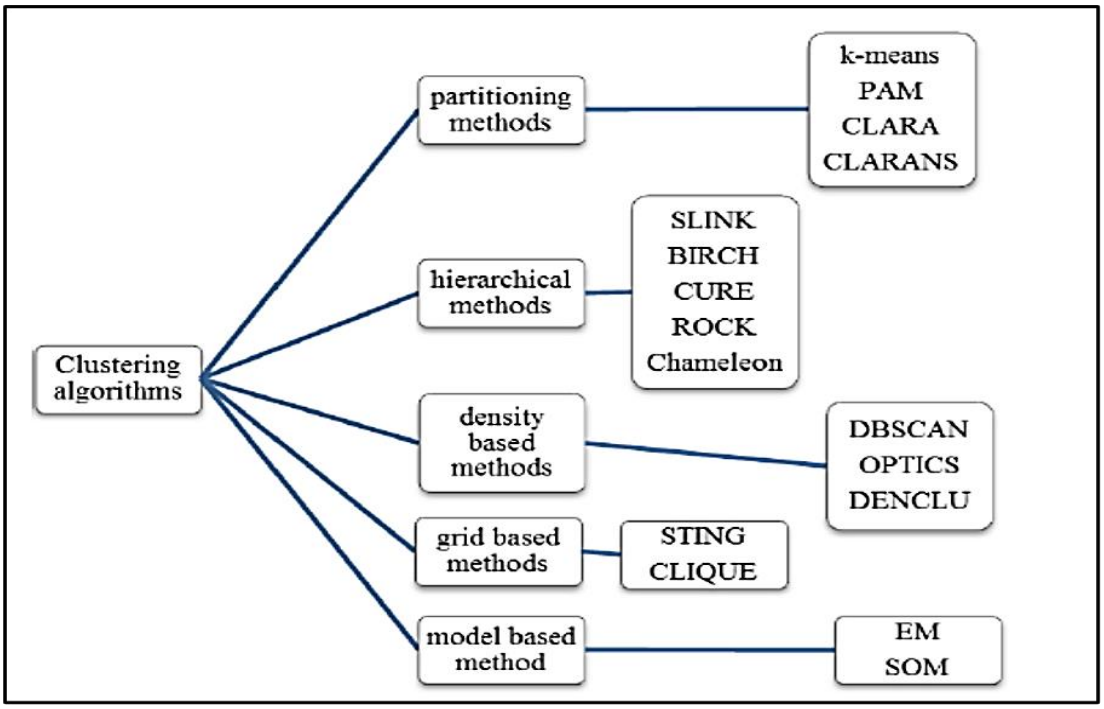


Figure (2.3) Classification of Clustering Algorithms

DBSCAN Clustering

DBSCAN (Density-Based Spatial Clustering of Application with Noise) is a technique for clustering data points into a number of partitioned clusters. DBSCAN is a type of density-based clustering approach that can identify arbitrarily shaped clusters, where dense regions are separated from lower-density regions to form clusters. The given clusters are developed further if the number of objects or data points within the adjacent area exceeds a predefined threshold [80].

There are two input parameters of DBSCAN are used to cluster data points [80]:

1. *MinPts*: Which represents a minimum number of points that must be included in a given cluster.
2. *Eps*: Which represents the radius of the cluster and is used to measure the distance between neighboring data points.

Two points are considered to be neighbors if the distance between them is less than or equal to *Eps*.

Based on these two input parameters, points are classified as core points, border points, or noise (outlier):

1. **Core point:** A point is a core point if there are at least *minPts* number of points (including the point itself) in its surrounding area with radius *Eps*.
2. **Border point:** A point is a border point if it can be reached from a core point and there are less than *minPts* number of points in its surrounding area.
3. **Noise (Outlier) point:** A point is a noise if it is not a core point and cannot be reached from any core point.

Figure (2.4) shows the types of points (core, border, outlier) in the DBSCAN algorithm, where A and C points are core points, B is border point, points N are noise points, the arrow represent *Eps*, *minpts*=4.

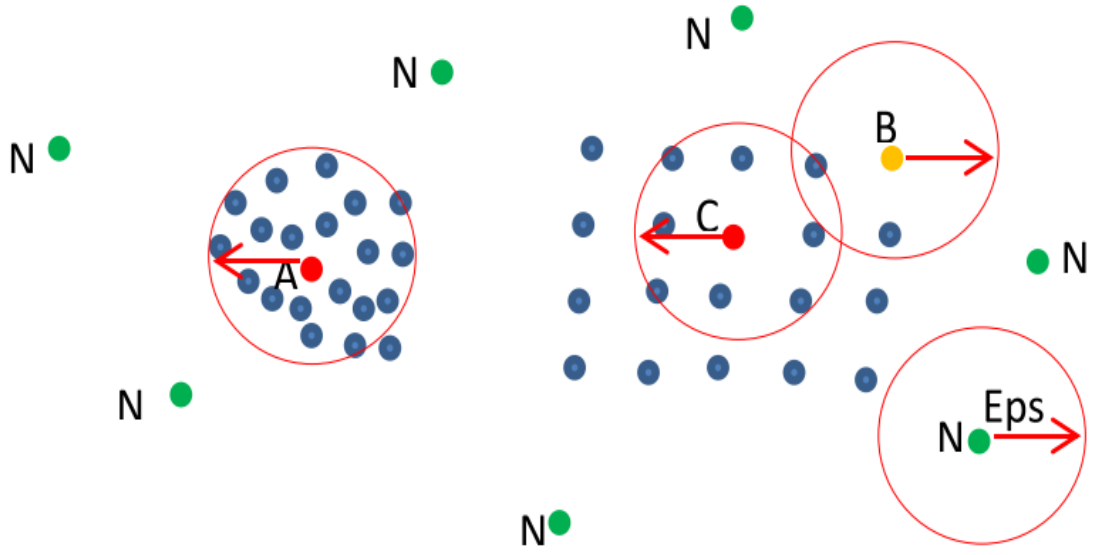


Figure (2.4) Types of Points in the DBSCAN Algorithm

The DBSCAN clustering algorithm is based on the following basic definitions [79]:

1. *Eps*-neighborhood of a point *x* denoted by $N_{Eps}(x) = \{y \in \text{Dataset} \mid dis(x,y) \leq Eps\}$.
2. Point *x* is directly density reachable from point *y* with respect to Eps and *minpts* if:
   a. $x \in N_{Eps}(y)$.
   b. $|N_{Eps}(y)| \geq minpts$
3. Point *x* is density reachable from *y* with respect to *Eps* and *minpts* if there exist a chain of points $z_1, z_2, \ldots, z_n$. Where $y=z_1$ and $x=z_n$, such that each point in the chain is direct density reachable from the previous one as shown in figure (2.5-a);
4. Point *x* is density connected to *y* with respect to *Eps* and *minpts* if there exist point *z* such that *x* and *y* are density reachable from *z* with respect to *Eps* and *minpts* as shown in figure (2.5-b) .
5. Cluster is non-empty subset of the input dataset with maximality and connectivity:
   a. If $x \in C$ and *y* is density reachable from *x* with respect to *Eps* and *minpts* then $y \in C$.
   b. If $x \in C$ and $y \in C$ then *x* is density connected to *y* with respect to *Eps* and *minpts*.
6. Noise is a set of points that are not belonging to any cluster. However, noise point does not belong to any neighborhood radius of core point and have number of points in its *Eps* radius less than *minpts*.

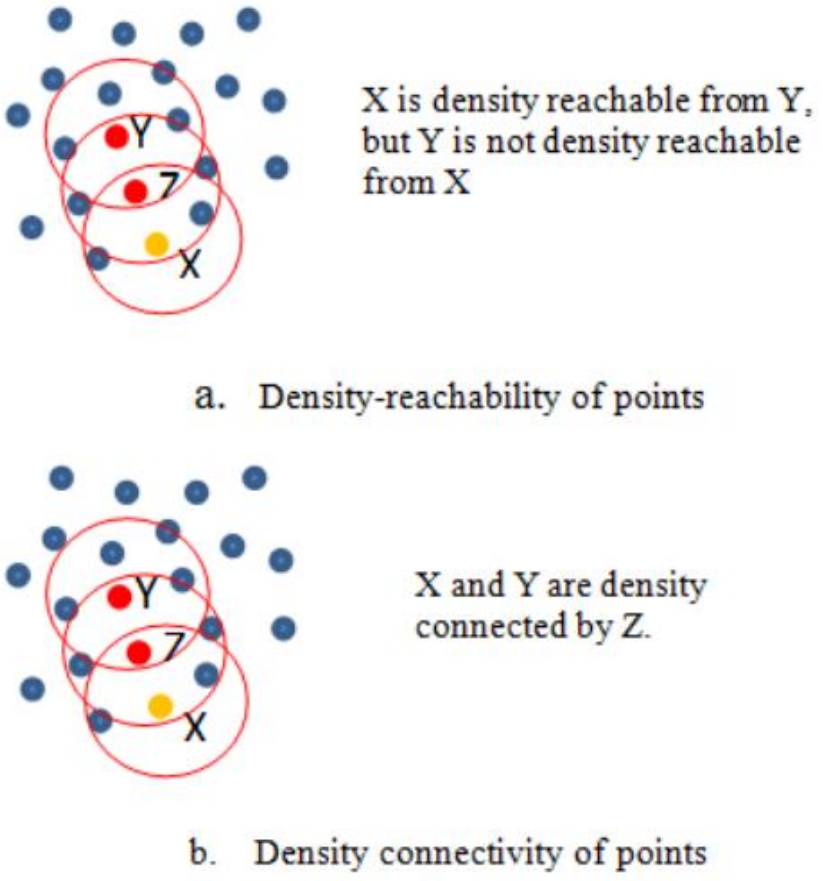


Figure (2.5) Density Reachability and Connectivity

The DBSCAN algorithm starts by picking up arbitrary point in the dataset that has not yet been visited, and checks the neighborhood based on a given value of Eps. If there is at least a minimum number of data points (*MinPts*) with an (*Eps*) to the current point, then all these points become part of the same cluster. Otherwise, that point will be marked as noise[79].

Most clustering methods use distance as a measure of the distance between two clusters, which fails to detect arbitrarily shaped clusters. DBSACAN can detect arbitrarily shaped clusters, which is the main feature of this technique in identifying clusters [81]. Algorithm (2.1) explain DBSCAN clustering [79].

Algorithm 2.1: DBSCAN Algorithm

```
DBSCAN(data,Eps,minpts)
Clus_id = 0
FOR I = 1 to size of data
 IF data[i] is unclassified THEN
   IF (|NEps(data[i])| ≥ minpts) THEN
      Clus_id = clus_id+1
      Expand_cluster(data[i], Eps, minpts, clus_id)
   ENDIF
 ENDIF
NEXT i
ALL unclassified points in data are noise
End DBSCAN
Expand_cluster(data, data[i], Eps, minpts, clus_id)
      Seed = data[i].regionquery(data[i],Eps)
      Data[i] and unclassified points in seed are
      assigned to clus_id
      Seed.delete(data[i])
      While seed <> empty do
            Point = Seed.getfirst
            Neighbor = point.regioquery(point,Eps)
            IF Neighbor.size ≥ minpts THEN
               Append all unclassified points in neighbor
               To seed and assign them to clus_id
            ENDIF
            Seed.delete(point)
      End while
End Expand_cluster
```

In Figure (2.6) an example about of DBSCAN clustering.

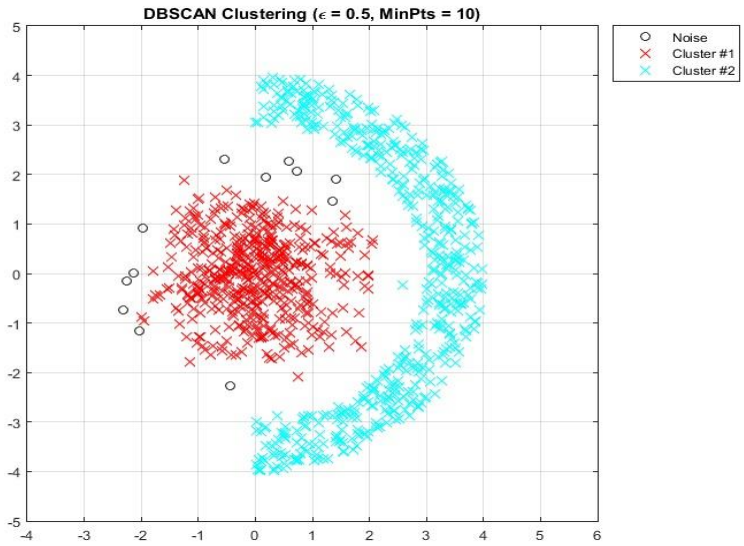


Figure (2.6) DBSCAN Clustering

2.8 Reputation and Trust System

On the Internet, users of services usually have inadequate information about the services offered, which can put users in vulnerable situations when purchasing these services. Thus, trust and reputation systems play an important role in decision making for Internet-mediated service delivery. The basic concept is that users submit their ratings about other users or for the services and then derive a reputation or trust score from the collected ratings about a particular user\service, which can help other users decide whether or not they want to deal with that user in the future [82].

The main differences between trust and reputation systems can be described as follows [83]:

1. Reputation can be viewed as a collective confidence measure in community which derived from member referrals or ratings of its members, while the subjective trust of an individual can be obtained from a received referrals as well as personal experience.

2. Reputation systems generate reputation score of an entity from the point of view of the entire community (public), while, trust systems generate a score that reflects the subjective opinion of the user on the trustworthiness of an entity (individual).

3. Transitivity is an explicit component in trust systems, whereas reputation systems usually consider transitivity only implicitly.

4. Trust systems usually use subjective and general measures of (reliability) trust as inputs, while information or ratings about specific (and objective) events, such as transactions, are used as inputs in reputation systems.

There is a similarity between collaborative filtering techniques and reputation systems, as both gather ratings from users. On the other hand, there is an important difference between them: Reputation systems assume that some users provide unreliable data for some reason, while collaborative filtering techniques assume that all users are equal in terms of credit and the values provided by them also reliable [83], [84].

2.8.1 Reputation Computation Methods

There are many methods for calculating reputation scores of entities in the Internet environment:

1. Traditional Computation Methods

In these methods, using a relatively simple statistical equations, In the simplest way, the reputation score can be calculated by summing up the number of positive and negative feedbacks separately and derive the total score as the positive feedbacks minus the negative feedback, an example of such a method is eBay's reputation system [85]. Another

method is to calculate the reputation score as the mean of all feedback, an example of such a method is Amazon and Epinions [86]. The main advantage of these methods is that they are easy to implement.

2. Dirichlet Reputation System

Dirichlet probability distribution, which is a multinomial Bayesian probability distribution. Dirichlet reputation systems represent a generalization of the binomial Beta reputation system. The multinomial aspect of the Dirichlet reputation system means that any set of discrete rating levels can be defined. This provides great flexibility and ease of use, as well as a solid foundation for designing reputation systems [87].

Multinomial Bayesian systems are based on computing reputation scores by statistically updating Dirichlet PDF. The a posteriori (i.e., updated) reputation score is calculated by combining the a priori (i.e., previous) reputation score with the new rating [88].

The probability distribution over disjoint elements of the multivariate state space are determined by the Dirichlet distribution, which describes the probability distribution over k-component random variable $p(\theta_i), i = 1 \dots k$ with sample space $[0,1]^k$, subject to the simple additivity requirement $\sum_{i=1}^{k} p(\theta_i) = 1$. [88]

The Dirichlet distribution captures a sequence of observations of the k possible outcomes with k positive real parameters $\alpha(\theta_i), i = 1 \dots k$ each corresponding to one of the possible outcomes. In order to have a compact notation, we define a vector $\vec{p} = \{p(\theta_i) \,|1 \leq\ i \leq k\}\ k$ to denote the k-component random probability variable and a vector $\vec{\alpha} = \{\alpha_i \,|1 \leq i \leq k\}$ to denote the k-component random observation variable $[\alpha(\theta_i)]_{i=1}^{k}$. Therefore, the Dirichlet probability function can be expressed as follows [88]:

$$\boldsymbol{f(\vec{p} \mid \vec{\alpha}) = \frac{\tau\left(\sum_{i=1}^{k} \alpha(\theta_i)\right)}{\prod_{i=1}^{k} \tau(\alpha(\theta_i))} \prod_{i=1}^{k} p(\theta_i)^{\alpha(\theta_i)-1}} \quad \ldots\ldots (2.9)$$

Where $\sum_{i=1}^{k} p(\theta_i) = 1, \quad p(\theta_1), \ldots, p(\theta_k) \geq 0$, $\alpha(\theta_1), \ldots, \alpha(\theta_k) > 0$, and τ is a gamma function subjective to $\tau(t) = \int_0^\infty x^{t-1} e^{-x} dx$.

The probability expectation value of the Dirichlet distribution is given by:

$$E(p(\theta_i) \mid \vec{\alpha}) = \left(\frac{\alpha(\theta_i)}{\sum_{i=1}^{K} \alpha(\theta_i)}, \right) \ldots\ldots\ldots\ldots (2.10)$$

Because of the additivity requirement $\sum_{i=1}^{k} p(\theta_i) = 1$,the Dirichlet distribution has only $k - 1$ degrees of freedom. This means that knowing $k-1$ probability variables and their density uniquely determines the last probability variable and its density.

2.9 Evaluation Strategies

Prediction accuracy is the most important criterion for evaluating QoS prediction of web services in collaborative filtering approaches. Two metrics are used, the first metric is the Mean of Absolute Error (MAE) and is obtained from the following equation:

$$\boldsymbol{MAE = \frac{1}{N} \sum_{i,j} |\bar{q}_{ij} - q_{ij}|} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.11)$$

Where $\boldsymbol{q_{ij}}$ is the actual value and $\boldsymbol{\bar{q}_{ij}}$ is the predicted value of QoS property, and $\boldsymbol{N}$ is the number of predicted QoS values.

Another metric used to measure predictive accuracy is the Root Mean Squared Error RMSE, and the following equation is used to calculate it:

$$RMSE = \frac{\sqrt{\sum_{i,j}\left(\bar{q}_{ij} - q_{ij}\right)^2}}{N} \qquad \ldots\ldots\ldots\ldots\ldots (2.12)$$

The lower values of these two criteria, the more accurate prediction we will have.

Chapter Three

The Proposed System

3.1 Introduction

This chapter clarifies how the proposed system is designed and implemented so as to build a personalized prediction for the QoS of non-functional values. It includes a description of the dataset that has been used in testing the system. As for the system itself, it consists of two stages, namely computing user reputation, and predicting unknown QoS values.

3.2 General Proposed System Architecture

Services are provided as shared resources in cloud computing, such as databases, software, and servers. Recently, as cloud computing has become more popular, an exponentially increasing number of cloud-based applications have emerged. The software architecture used to deliver multiple cloud services in a cloud environment is typically a web service. In the real world, the QoS of cloud services is strongly affected by the network environment and the location of the users. Moreover, only a limited number of these services are invoked by the service users. Therefore, predicting the QoS values of cloud services becomes an urgent and critical challenge in cloud services.

In this work of dissertation, two factors of contextual information were considered in building a proposed CF-based approach for predicting unknown QoS values: User reputation and geographical location. User reputation was used to identify untrusted users and fix their unreliable data. This ensures that the data contributed by users is credible, as well as reducing the impact of the data sparsity problem by correcting unreliable data. While geographic location has been used to improve similarity computation to improve the accuracy of QoS prediction.

Figure (3.1) shows the main parts of proposed system of the proposed work for Cf-based approach based on reputation and location-based. The Input data part is responsible for extracting, transforming and loading information from the sources. The data is divided into two types, namely QoS data and user locations.

In the user reputation part, to identify untrusted users in the QoS-dataset, preprocessing of QoS data is performed on the QoS data matrix, and then user reputation calculation is implemented using Dirichlet reputation system. Finally, the unreliable QoS data is processed to create a new QoS matrix.

In the QoS prediction part, to determine the neighbors of the users, a clustering process has been implemented based on the geographical location of the users. Then, the similarities between neighboring users in each cluster using cosine similarity are calculated according to the new QoS matrix. Finally, unknown QoS values are predicted based on the similarity of neighboring users.

This work uses a real-word dataset proposed by Zheng at al.[89] and available as WS-Dream. It contains four matrices: a user-list matrix that contains context information about 339 users, a WS-list matrix that contains context information about 5825 web services, and two other matrices that contain QoS data of web services., one showing response time data and the other showing throughput data of 5825 web services used by 339 users. The dataset contains 1,974,675 records of response time and throughput attributes.

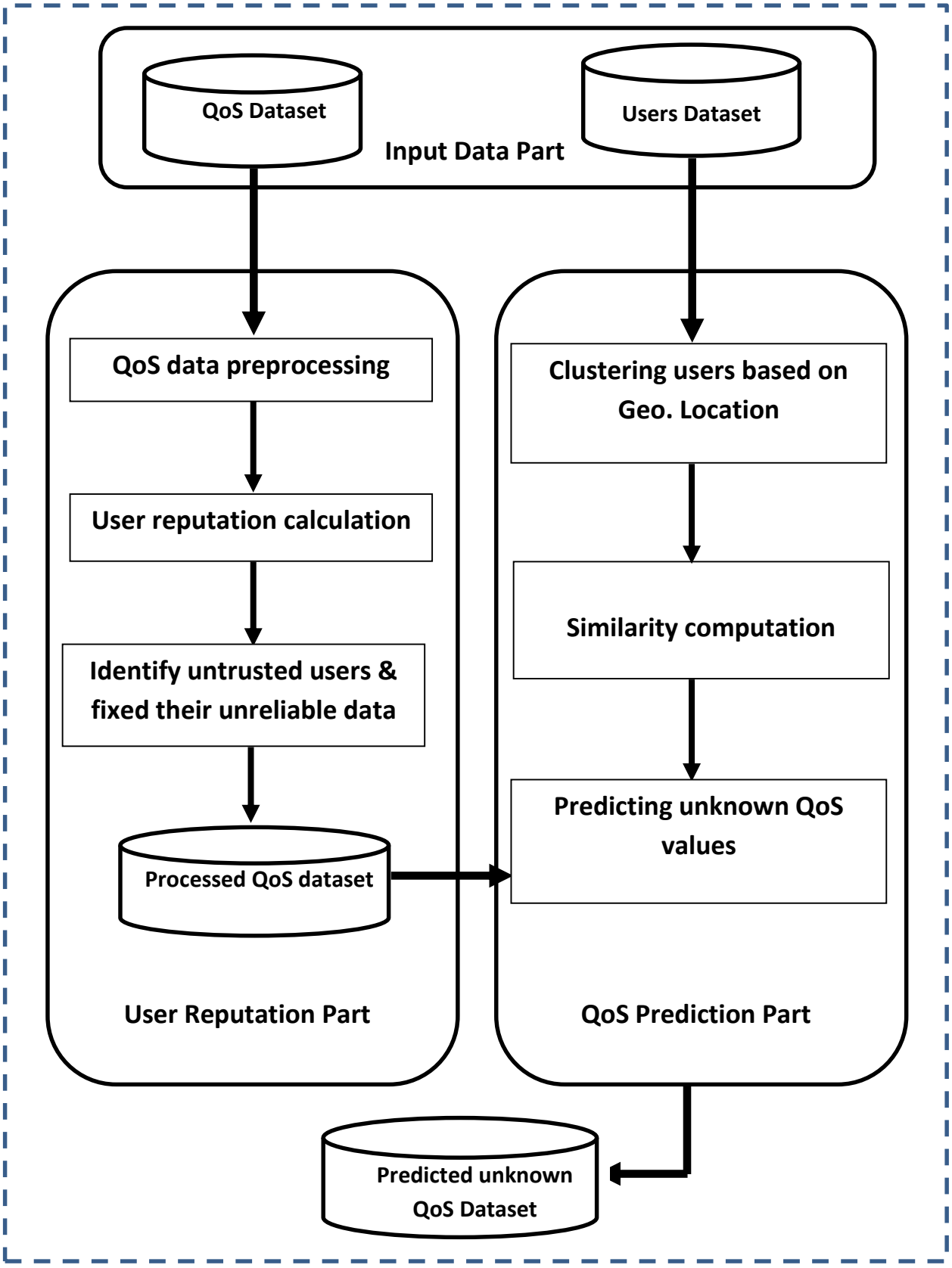


Figure (3.1) Proposed System Diagram

3.3 The Process of Personalized QoS Prediction

As shown in Figure (3.1), the process of the proposed approach for personalized QoS prediction consists of two main parts: 1) computing user' reputation part, where the QoS data obtained from different users is used to detect untrusted users. 2) predicting unknown QoS values part, where the missing QoS values are predicted by using reliable QoS values based on the similarity degree of all neighbors.

3.4 User Reputation-based Calculation

Non-functional QoS attributes are the ones whose values depend on other parameters and can change, including response time and throughput which are recorded and evaluated on the user-side. Therefore, the dynamic network environment leads to different QoS value observations made by different users while using the same services. For some reason, some of these data may be unreliable. Normally, the QoS values observations made by the majority of users should be part of the normal range, whereas any significantly deviating observations from this normal range are not likely to occur. Thus, the assumption is made that whenever users consistently submit QoS feedbacks that deviate significantly from the majority of users, then those users are not to be trusted. Consequently, the probability of the user's trustworthiness can be evaluated using user feedback information from historical invocation values.

3.4.1 QoS Data Pre-processing

The QoS data obtained during the invocation of web services by users are processed first. Since the QoS data is very different for all users of each web service, the QoS value of each service must be normalized to

a reasonable range to better partition the statistical intervals. The QoS values are normalized using the most common method of minimum and maximum normalization, often between 0 and 1. Each QoS value should be normalized as shown in the following Equation [90]:

$$NR_{ij} = \frac{R_{ij} - R_j^{min}}{R_j^{max} - R_j^{min}} \quad \text{..... (3.1)}$$

Where NR_{ij} is the normalized value of the original R_{ij}, R_j^{min} and R_j^{max} represent the minimum and maximum QoS values for service s_j, respectively. After normalization, all QoS values are scaled between [0, 1].

Algorithm (3.1) has shown how to normalize QoS value of QoS matrix.

Algorithm 3.1: Normalize QoS Values	
Input: QoS Dataset	
Output: Normalized QoS Dataset	
Begin	
1.	***for*** *j=1,2,3,...,M* ***do*** / M no. of services
2.	R_j^{min} ← ***Min value*** / Find minimum value of invocation for service j
3.	R_j^{max} ← ***Max value*** / Find maximum value of invocation. for service j
4.	***for*** *i=1,2,3,...,N* ***do*** / N no. of users
5.	NR_{ij} ← normalize QoS values by Eq. (3.1)
6.	*End for j*
7.	*End for j*
8.	**return NR** / Normalized QoS Dataset
End algorithm	

3.4.2 Determining Feedback Vector

In this section, the user feedback information is classified to trusted, untrusted, and no feedback. The feedback vector for each user in user-service matrix is obtained by first finding a set of users in each service that represent the reliable group, and then count the feedback information for each user based on trusted and untrusted feedback corresponding to the reliable group in each service. To find reliable user groups, the statistical interval division method is used to divide the normalized QoS data of all users for each web service within user-service matrix. For example, the normalized QoS value in [0, 1] intervals can be evenly divided into 11 intervals of (0, 0.1], (0.1, 0.2], ..., (0.9, 1], and [0, 0].

Since reliable users often represent the majority of users and their QoS data is within the normal range. Therefore, intervals with the majority of users are considered reliable. The user group in the reliable interval on the service s_j is defined as follows:

$$\boldsymbol{u_j^{max} = \{u | u \in G_j^l, l = argmax_k \left|G_j^k\right|} \quad \text{..... (3.2)}$$

Where $\left|\boldsymbol{G_j^k}\right|$ is the set of users in the kth groups, l represents the index of the group with the greatest number of users, $\boldsymbol{u_j^{max}}$ reflects the number of users in reliable group on the service s_j. As stated previously, QoS values that deviates highly from normal is unlikely, thus by assessing the QoS values introduced by the user and the deviation to the reliable group, the user feedbacks are classified into trusted and untrusted feedbacks. Trusted feedback refers to the QoS values is in the majority, otherwise it is untrusted feedback.

It is assumed that the quality of service (QoS) values for web services, contributed by various users, follow the Gaussian distribution $G(\mu, \sigma^2)$ with μ and σ being the average and standard deviation of reliable group in each service, respectively. Given the 3σ of Gaussian normal distribution [91], the probability that users observe QoS values in the range $[\mu - 3\sigma, \mu + 3\sigma]$ is very close to 99.7%, and thus the probability P of the QoS values of service s_j according to the observation of the user u_i is expressed as follows:

$$\boldsymbol{P\left(\mu_j^l - 3\sigma_j^l < NR_{ij} \leq \mu_j^l + 3\sigma_j^l\right) = 99.7\%)} \quad \text{…….. (3.3)}$$

Where μ_j^l and σ_j^l is the mean and standard deviation of the reliable group l of service s_j, respectively, and NR_{ij} is the normalized QoS value of service s_j observed by user u_i.

According to Equation 3.3, a normalized user feedback NR_{ij} can be classified as being either trusted, untrusted, or no feedback, as follows:

$$\boldsymbol{NR_{ij} = \begin{cases} Trusted\ feedback, & if\ \left|NR_{ij} - \mu_j^l\right| \leq 3\sigma_j^l \\ Untrusted\ feedback, & if\ \left|NR_{ij} - \mu_j^l\right| > 3\sigma_j^l \\ no\ feedback & if\ NR_{ij} = 0 \end{cases}} \quad \text{…….. (3.4)}$$

After classifying all user feedback information for each user, the feedback information can be counted and expressed as a feedback vector:

$$\overrightarrow{Fd_i} = [pt_i, pu_i, no_i] \quad \text{……..…. (3.5)}$$

Where $\overrightarrow{Fd_i}$ is the feedback vector of user i, pt_i is the number of trusted feedbacks of user i, pu_i is the number of untrusted feedbacks of user i, and no_i is the number of times without feedback.

Algorithm (3.2) has shown how to calculate users' feedback vector.

Algorithm 3.2: Calculate User's Feedback Vector

Input: **Normalizes QoS Dataset, Users list, Services List**

Output: **Feedback vector**

Begin

1. ***for*** *i=1,2,3,...,N* ***do*** / N no. of users

2. $pt_i , pu_i , no_i \leftarrow 0$ / initialize Trusted, Untrusted and No Feedback values for user $_i$

3. *End for i*

4. ***for*** *j=1,2,3,...,M* ***do*** / M no. of services

5. divide normalized QoS values in each service into statistical intervals

6. find the reliable interval with the maximum number of users according to Eq. (3.2)

7. ***for*** *i=1,2,3,...,N* ***do*** */ N no. of users*

8. ***if*** *no feedback for* $service_j$ *observed by* $user_i$ ***then***

9. $no_i + 1 \leftarrow no_i$ / no_i no. of no feedback

10. ***else***

11. find mean (μ) and standard deviation (σ) of reliable interval for $service_j$

12. ***if*** $|NR_{ij} - \mu| \leq 3\sigma$ ***then*** / NR_{ij} normalized QoS values

13. pt_i+1 $\leftarrow pt_i$ / pt_i no. of Trusted feedback

14. **else**

15. $pu_i + 1 \leftarrow pu_i$ */ pu_i no. of untrusted feedback*

16. *End_if*

17. *End_if*

18. *End for i*

19. *End for j*

20. ***for*** *i=1,2,3,...,N* ***do*** */ N no. of users*

21. $\overrightarrow{Fd_i} \leftarrow [pt_i, pu_i, no_i]$ / $\overrightarrow{Fd_i}$ Feedback vector for user $_i$

22. *End for i*

23. **return** $\overrightarrow{Fd}$ / Feedback vector

End algorithm

3.4.3 Calculating User' Reputation

After determining the user feedback vector, the next step is to use the user reputation calculation to identify untrustworthy users. A user's reputation can be viewed as measure of trustworthiness of the community based on a user's historical feedback behaviour. Consequently, calculating a user's reputation can provide motivation for trustworthy feedback behaviour while helping users determine who is a credible user. In the proposed approach, the Dirichlet probability distribution is used for calculate a user's reputation based on the user's feedback vector.

As described earlier, suppose that the observed feedback $\overrightarrow{Fd_i}$ from user u_i contains pt_i trusted feedback, pu_i untrusted feedback, and no_i no feedback, therefore the vector $\vec{m} = (pt_i, pu_i, no_i)$ represents the count vector. According to the Dirichlet probability function, the feedbacks probability $\vec{p}$ of user u_i in the future can be expressed as follows:

$$f(\vec{p}|\vec{1} + \vec{m}) = \frac{\tau(pt_i+pu_i+no_i+3)}{\tau(pt_i+1)\,\tau(pu_i+1)\tau(no_i+1)} \cdot \left(p_{i,1}^{pt_i}\ p_{i,2}^{pu_i}\ p_{i,3}^{no_i}\right) \quad \ldots. (3.6)$$

The probability expectation of the Dirichlet distribution is most likely the user's probability value $\vec{p}$. Thus, once the various user feedback is identified, the probability of trusted, untrusted, and non-feedbacks of future invocation services for the user u_i could be given by the following:

$$p_{i,1} = E(f(\vec{p}\,|\vec{1} + \vec{m}_i) = \frac{pt_i+1}{pt_i+pu_i+no_i+3}$$

$$p_{i,2} = E(f(\vec{p}\,|\vec{1} + \vec{m}_i) = \frac{pu_i+1}{pt_i+pu_i+no_i+3} \quad \ldots\ldots\ldots (3.7)$$

$$p_{i,3} = E(f(\vec{p}\,|\vec{1} + \vec{m}_i) = \frac{no_i+1}{pt_i+pu_i+no_i+3}$$

Where $p_{i,1}$, $p_{i,2}$, and $p_{i,3}$ represent the probability of trusted, untrusted, and non-feedbacks for future invoking services by user u_i, respectively. The user's reputation is calculated only after the user invoking the service and receiving feedback. Thus, the user's reputation is considered to be the probability of trusted feedback whenever they receive feedbacks, and it can express as follows:

$$\boldsymbol{Rep_i = \frac{p_{i,1}}{p_{i,1}+p_{i,2}}, \quad (p_{i,1}+ p_{i,2}) \neq 0} \qquad \text{..... (3.8)}$$

Where Rep_i is the reputation of the user u_i, which is in the range of [0,1].

3.4.4 Identifying Untrusted Users and Fixed their Unreliable Data

After calculating user's reputation for all users according to the Dirichlet probability distribution and the QoS value of the web services observed by the user. The untrusted user is identified according to the user reputation value by setting a trusted threshold. If the user's reputation value is below the threshold δ, then the user is considered to be untrusted. The identification of untrusted users could be described in the following way:

$$\boldsymbol{\bar{u} = \{u | Rep_u < \delta\}} \qquad \text{..... (3.9)}$$

Where $\bar{u}$ represents the set of users identified as untrusted and δ represents the threshold. Obviously, if the unreliable data for untrusted users is removed in a direct manner, the user-service matrix becomes sparser, which greatly affects how accurate the QoS predictions are. To mitigate the sparse data problem, the unreliable QoS value submitted by untrusted users can be dealt with using the following equation:

$$r'_{u,j} = \begin{cases} \overline{Ru_j^{mean}} & , r_{u,j} > 0 \ and \ r_{uj} > \overline{Ru_j^{mean}} \\ 0 & , r_{u,j} = 0 \end{cases} \text{, and } u \in \bar{u} \quad \ldots. (3.10)$$

Where $r'_{u,j}$ is the modified QoS value of untrusted user u on service s_j, $r_{u,j}$ represent the original QoS value of that user u in service s_j, and $\overline{Ru_j^{mean}}$ represents the average QoS values in the reliable set of service s_j.

Algorithm (3.3) has explained how to calculate the user's reputation using Dirichlet Reputation system.

Algorithm 3.3: Calculate Users' Reputation		
Input: QoS Dataset, User's Feedback Vector		
Output: Reliable QoS Values		
Begin		
1.	***for*** *i=1,2,3,…..,N* ***do***	*/ N no. of users*
2.	$p_{i,1} \leftarrow \frac{pt_i+1}{pt_i+pu_i+no_i+3}$	/ $p_{i,1}$ the probabilities of trusted feedback of user u_i
3.	$p_{i,2} \leftarrow \frac{pu_i+1}{pt_i+pu_i+no_i+3}$	/ $p_{i,2}$ the probabilities of untrusted feedback of user u_i
	$p_{i,3} \leftarrow \frac{no_i+1}{pt_i+pu_i+no_i+3}$	/ $p_{i,2}$ the probabilities of no feedback of user u_i
4.	$Rep_i \leftarrow \frac{p_{i,1}}{p_{i,1}+p_{i2}}$	/ Rep_i the reputation of user u_i
5.	***If*** $Reb_i < \delta$ ***then***	/ δ threshold and equal to 0.5
6.	$\bar{u} \leftarrow u_i$	/ $\bar{u}$ untrusted users
7.	*End_if*	
8.	*End for i*	
9.	//Fixing unreliable QoS values	

10.	***for*** $u=1,2,3,\ldots,Q$ ***do*** / Q no. of untrusted users
11.	***for j =*** $1,2,3,\ldots,M$ ***do*** / M no. of services
12.	***If*** $r_{uj} > 0$ ***and*** $r_{uj} > Ru_j^{mean}$ ***then***
13.	$r'_{u,j} \leftarrow Ru_j^{mean}$ / Ru_j^{mean} mean of the QoS values in reliable interval of service j
14.	*End_if*
15.	*End for j*
16.	*End for u*
17	**return Reliable QoS values**
End algorithm	

3.5 Prediction Unknown QoS Values

QoS values are much determined by the performance of underlying networks. Therefore, the users located at the same location tend to present the same QoS attribute values on the service itself due to the similar infrastructures and the distances that they share to the service provider. Users with differing locations tend to observe different QoS values for the same services. Thus, adding the user's geographic location as context in a CF-based approach improves the accuracy of predicting missing QoS values and also mitigates the impact of data sparsity problem.

First, the DBSCAN clustering algorithm is applied to the user sides for forming clusters to reveal the geographical similarity between users. By clustering users according to their geographical location, the data volume is reduced and reduces the computational overhead. As mentioned before, the users in the same cluster tend to have QoS values that are alike.

There are two cases for predicting the unknown QoS value of service s_j for user u_i: (1) the service s_j was previously invoked by other

similar users in the same cluster. (2) the service s_j has not been invoked before by any user in the same cluster.

Algorithm (3.4) has shown how to predict the unknown QoS values.

Algorithm 3.4: Unknown QoS Values Prediction	
Input:	**U: User matrix** **R: Reliable QoS Matrix**
Output:	**R^*: Predicted Unknown QoS Values**
Begin	
1.	// Remove services that have not invocation by all users
2.	***for*** *i=1,2,3, …., M* ***do*** / M no. of services in the QoS matrix
3.	***if*** $S_i = \phi$ **then**
4.	*remove (S_i)*
5.	*End_if*
6.	*End for i*
7.	// Clustering users based on their geographical location
8.	C ← DBSCAN (minpts, Eps) / minpts: minimum no. of points /Eps: radius of the cluster
9.	// Predicting Unknown QoS Values
10.	***for*** *x=1,2,3, …., T1* ***do*** / T1 no. of clusters
11.	***for*** *i=1,2,3, …., T2 do* / T2 no. of users in cluster x
12.	***for*** *j=1,2,3, …., Q* ***do*** / Q no. of services that not previously invoked by the user i
13.	***if*** "there are neighbors who have previously invoked the service j" ***then***
14.	*predict R_{ij}^* in the same cluster x by Eq. (3.13)*
15.	*else predict R_{ij}^* in the closest cluster to cluster x by Eq. (3.16)*
16	*End_if*
17.	*End for j*
18.	*End for i*
19.	*End for x*
20.	**return** R^* **(Predicted Unknown QoS Values)**
End algorithm	

3.5.1 Predicting QoS Values in the Same Cluster

In the first case, the symbol φ_{ik} is used to refer to the set of neighbor users of user u_i in the same cluster which previously have observed QoS values for service s_k:

$$\boldsymbol{\varphi_{ik} = \{u_j | u_j \in C_n, R_{jk} > 0\}} \quad \text{..... (3.11)}$$

Where C_n represents the set of users within cluster n, R_{jk} denotes the QoS value of service s_k according to the observation of user u_j.

After all users in the dataset have been assigned clusters, the similarity between different users in each cluster can be calculated through the deployment of the cosine similarity. The following equation defines the similarity between different users:

$$\mathbf{Sim}(\boldsymbol{u_i, u_j}) = \frac{\sum_{k \in S_{ij}} \boldsymbol{R_{ik} R_{jk}}}{\sqrt{\sum_{k \in S_{ij}} (\boldsymbol{R_{ik}})^2} \sqrt{\sum_{k \in S_{ij}} (\boldsymbol{R_{jk}})^2}} \quad \text{..... (3.12)}$$

Where $Sim(u_i, u_j)$ denotes the similarity between user u_i and u_j. $S_{ij} = S_i \cap S_j$ represents the set of previously invoked services by both users u_i and u_j. R_{ik} represent the QoS value of service S_k according to the observation of user u_i. R_{jk} represent the QoS value of service S_k according to the observation of user u_j.

Then, the missing QoS value of service s_k for user u_i can be predicted by the following equation:

$$\boldsymbol{R_{ik} = \sum_{k \in \varphi_{ik}} \frac{Sim(u_i, u_j)}{\sum_{k \in \varphi_{ik}} Sim(u_i, u_j)}} R_{jk} \quad \text{..... (3.13)}$$

Where R_{ik} denotes the predicted value for s_k observed by u_i, R_{jk} denotes the QoS value of user u_j who has previously invoked service s_k.

Algorithm (3.5) shows predicting unknown QoS values in the same cluster.

Algorithm 3.5: Predicting Unknown QoS Values in the Same Cluster	
Input:	**C: User cluster** **R: Reliable QoS Matrix**
Output:	**R^* : Predicted Unknown QoS Values**
Begin	
1.	***for*** *i=1,2,3, …., N* ***do*** / N no. of users in cluster C that do not make previous invocation for some services.
2.	***for*** *k=1,2,3, …., M* ***do*** / M no. of services not invoked by user i.
3.	φ_{ik} ← *Construct user neighbor set for user i who invoked service k by Eq. (3.11)*
4.	***if*** $\varphi_{ik} \neq \emptyset$ *then*
5.	// calculate the similarity between users
6.	***for*** *j=1,2,3, …., T* ***do*** / T no. of users who have invoked services k.
7.	$\boldsymbol{Sim(u_i, u_j)}$ ← calculate the similarity by Eq. (3.12)
8.	*End for j*
9.	// predicting unknown QoS values
10.	***for*** *j=1,2,3, …., T do* / T no. of users who have invoked services k.
11.	$\boldsymbol{R^*_{ik}}$ ← predict the unknown QoS values by Eq. (3.13)
12.	*End for j*
13.	*End_if*
14.	*End for k*
15.	*End for i*
16.	**return R^*_{ij} (Predicted Unknown QoS Values)**
End algorithm	

3.5.2 Predicting QoS Values in the Closest Cluster

In the second case, if the neighbors of user u_i in the same cluster have not invoked service s_k before, the QoS values of user u_i can be observed in the closest neighbor cluster. Therefore, the distance between the user u_i and its cluster center c_i is first calculated with the centers of the other clusters. The distance between clusters can be obtained by the following equation:

$$\boldsymbol{d_{ij} = \frac{d(u_{i,}c_j)+d(c_i,c_j)}{2}} \quad \text{..... (3.14)}$$

Where $d(u_{i,}c_j)$ and $d(c_i, c_j)$ are the Euclidean distance between u_i and its cluster center c_i to the center c_j respectively.

The definition of the closest cluster for user u_i as d^{min} could be sated as follows:

$$\boldsymbol{d^{min} = d\left|d \epsilon D^i, i = argmin_k\right|D^k|} \quad \text{..... (3.15)}$$

Where $|D^k|$ denotes the distance of the kth cluster, i represent the index of the cluster with the smallest distance to the user u_i and its cluster center.

After assigning the closest group, the missing QoS value of user u_i for service s_k in cluster c is predicted according to the following equation:

$$\boldsymbol{R_{ik}^c = \frac{\sum_{K\epsilon\varphi_{ik}} R_{jk}^t}{N}} \quad \text{..... (3.16)}$$

Where R_{jk}^t denote to observed values of users who have invocation for service s_k in cluster t , N represent the number of users who have invocation for service s_k in the nearest cluster t.

If there are no users that have invoked the service s_k in the first closest cluster, the prediction of the QoS value of the service s_k observed by the user u_i is calculated in the next closest cluster.

Algorithm (3.6) shows predicting unknown QoS in the closest cluster.

Algorithm 3.6: Predicting Unknown QoS Values in Closest Cluster	
Input:	**C_s: Set of Clusters** **C : Current cluster** **R: Reliable QoS Matrix**
Output:	**R^c: Predicted Unknown QoS Values**
Begin	
1.	***for*** *i=1,2,3, …., N* ***do*** / N no. of users in cluster C that do not make Previous invocation for some services.
2.	***for*** *k=1,2,3, …., M* ***do*** / M no. of services not invoked by user i.
3.	φ_{ik} ← Construct user neighbor set for user i who invoked service k
4.	***if*** $\varphi_{ik} = \emptyset$ ***then***
5.	// calculate the distance between current cluster c_i for user i and other clusters
6.	***for*** *j=1,2,3, …., T1* ***do*** /T1 no. of clusters (C_s)
7.	c_i ← center of current cluster for user i
8.	c_j ← center of cluster j
9.	$\boldsymbol{d_{ij}}$ ← calculate the distance between location of user I and its cluster center j from another cluster by Eq.(3,14)
10.	*End for j*
11.	$t \leftarrow indexmin(d)$ / Find minimum distance
12.	φ_{ik}^t ← Construct user set for user i who invoked service k in closest cluster
13.	***if*** $\varphi_{ik}^t \neq \emptyset$ ***then***
14.	// predicting unknown QoS values in the closest cluster
15.	***for*** *j=1,2,3, …., T2* ***do*** T2 no. of users who have invoked services k in the closest cluster t
16.	$\boldsymbol{R_{ik}^c}$ ← predict the unknown QoS values by Eq. (3.16)

17.	*End for j*
18.	*End_if*
	End _if
19.	*End for k*
20.	*End for i*
21.	**return R^c (Predicted Unknown QoS Values)**
End algorithm	

Chapter Four

Implementation and Results

4.1 Introduction

In this section, we discuss the results of the experiment for the proposed approach using the real dataset to test the results and performance of the system available as WS-Dream that proposed and collected by Zheng et al [89]. The proposed system is applied on computer which meets particular criteria, among which are :

- Hardware, Processor: Intel(R) Core (TM) i7-8565U CPU @ 1.8GH_z (8CPUs). Memory: 8.00 GB RAM.
- Operating System: Windows 10 Pro 64-bit.
- Programming Language: Python 3.8

4.2 Dataset Description

In the real world, it is very expensive to invoke thousands of web services for large scale experiments. To evaluate our proposed QoS prediction approach, the experiments were conducted using the WS-Dream dataset proposed and collected by Zheng [89], which collects the QoS data of web service invocations and the context information of different users. Figure (4.1) shows how the dataset is collected.

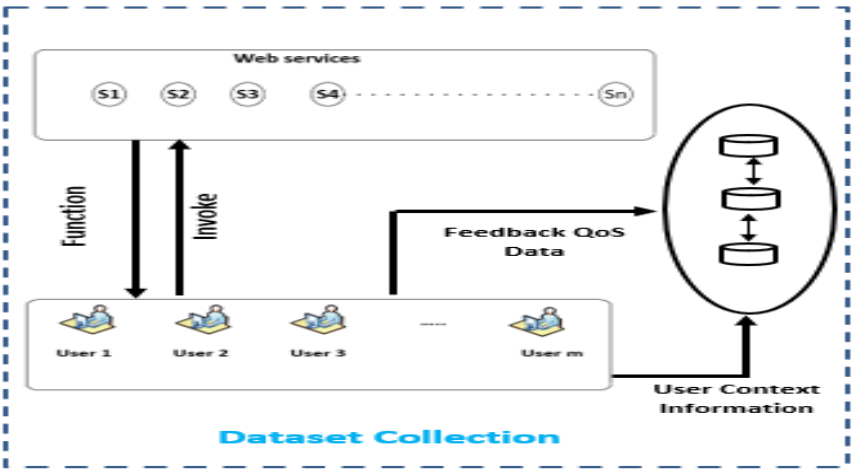


Figure (4.1) QoS Data Collection

By processing the Web service invocation results, two 339 × 5,825 matrices were created for response time attribute and throughput attribute, respectively. Each entry in a matrix represents the response-time value or throughput value observed by a user on a Web service. In each matrix, there exist 1,974,675 QoS records for 5825 web services distributed in 73 countries and observed by 339 users from 30 countries. Figure (4.2) shows the two matrices.

2.337	0.158	0.108	2.283	0.425	0.264	2.294
-1	1.007	0.715	0.616	0.527	0.263	0.581
0.18	0.161	0.207	0.161	0.161	0.268	0.207
0.428	7.026	-1	0.363	0.184	-1	-1
0.101	0.195	0.404	0.678	0.442	0.102	1.025
1.202	8.046	1.993	4.795	4.394	4.7	1.308
5.212	1.244	1.209	1.172	0.89	0.91	3.451
0.904	14.701	1.52	1.44	3.676	1.721	1.528
0.296	1.88	0.869	1.611	0.588	2.975	0.306
-1	-1	-1	-1	-1	-1	-1
0.292	0.301	0.292	0.296	0.294	0.294	0.294
0.056	0.172	5.473	0.828	1.031	1.272	0.95
0.273	0.839	0.282	0.896	0.306	0.306	1.176
0.822	0.16	0.675	3.671	0.659	0.129	0.151
0.648	0.648	0.886	10.669	-1	-1	3.399
0.236	1.455	0.207	2.64	0.358	0.246	0.243
0.282	0.282	0.281	0.281	0.276	0.283	0.281

(a) Response time matrix

62.999	12.765	11.764	7.594	63.829	1.112	0.379	14.388
9.925	7.633	14.981	0.45	0.604	0.334	7.722	0.492
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
0.388	21.276	1.164	20.689	15.306	24.59	0.524	24.793
4.962	0.337	5.555	0.331	5.464	2.244	5.747	0.551
366.013	6.972	363.636	0.343	0.882	87.719	0.523	0.532
790.419	440.944	-1	522.058	827.777	642.857	476.19	-1
0.332	0.526	173.553	22.727	27.027	2.573	13.26	1.664
0.53	25.0	0.536	25.0	25.21	2.518	2.066	0.5
0.515	12.658	0.468	0.716	1.864	0.52	0.65	0.517
21.276	11.764	21.582	0.69	250.0	-1	15.873	1.078
28.169	0.362	0.512	0.546	0.344	21.546	21.438	4.518
49.019	45.871	37.037	40.65	46.296	40.0	49.019	46.296
159.292	20.408	10.309	9.039	966.666	634.146	-1	383.177
6.185	6.835	0.518	15.625	0.528	66.666	-1	15.384
6.42	24.193	24.0	38.961	122.302	121.428	8.347	22.9
-1	-1	-1	-1	-1	-1	806.451	-1
187.5	7.009	30.303	48.387	47.619	10.714	52.631	63.829

(b) Throughput matrix

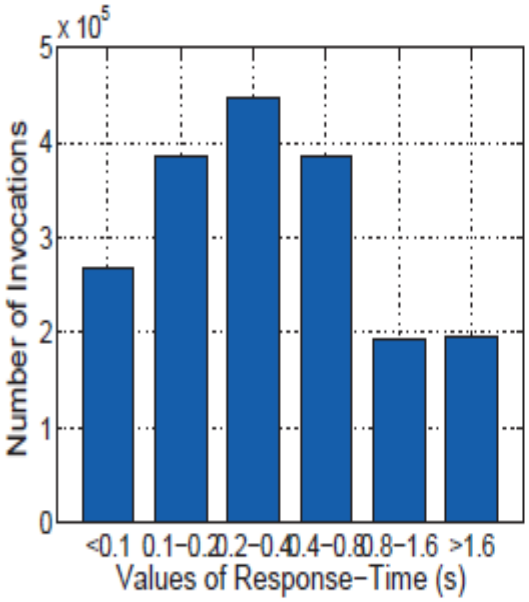
Figure (4.2) QoS Attribute Matrices

WS -Dream dataset contains two other files that represent the user profile and the service profile. The user profile consists of the user's context information and includes seven attributes that describe basic information for users: User ID, IP Address, Country, IP Number, Autonomous System (AS), Latitude, and Longitude. While The service profile consists of the service's context information and includes nine attributes that describe basic information for services: Service ID, WSDL Address, Service Provider, IP Address, Country, IP Number, Autonomous System (AS), Latitude, and Longitude. Figure (4.3) shows the user profile matrix.

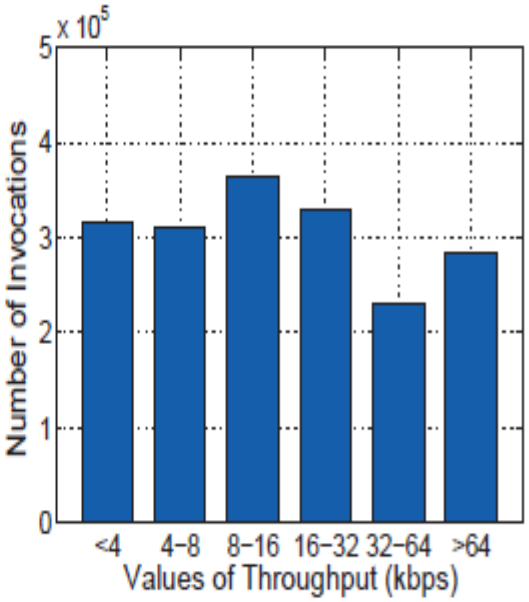
[User ID]	[IP Address]	[Country]	[IP No.]	[AS]	[Latitude]	[Longitude]
67	129.242.19.197	Norway	2180125637	AS224 UNINETT, The Norwegian University & Research Network	69.6667	18.9667
68	129.59.88.180	United States	2168150196	AS7212 Vanderbilt University	36.1473	-86.777
69	129.69.210.96	Germany	2168836704	AS553 Landeshochschulnetz Baden-Wuerttemberg (BelWue)	48.7667	9.1833
70	129.69.210.97	Germany	2168836705	AS553 Landeshochschulnetz Baden-Wuerttemberg (BelWue)	48.7667	9.1833
71	129.74.74.19	United States	2169129491	AS693 University of Notre Dame	41.7007	-86.2501
72	129.74.74.20	United States	2169129492	AS693 University of Notre Dame	41.7007	-86.2501
73	129.93.229.138	United States	2170414474	AS7896 University of Nebraska-Lincoln	40.8	-96.667
74	129.93.229.139	United States	2170414475	AS7896 University of Nebraska-Lincoln	40.8	-96.667
75	129.97.74.12	Canada	2170636812	AS12093 University of Waterloo	43.4529	-80.5281
76	129.97.74.14	Canada	2170636814	AS12093 University of Waterloo	43.4529	-80.5281
77	130.136.254.21	Italy	2190016021	AS137 GARR Italian academic and research network	44.4833	11.3333
78	130.149.49.136	Germany	2190815624	AS680 service G-WiN	52.5167	13.4
79	130.149.49.137	Germany	2190815625	AS680 service G-WiN	52.5167	13.4
80	130.192.86.30	Italy	2193643038	AS137 GARR Italian academic and research network	45.05	7.6667
81	130.206.158.140	Spain	2194579084	AS766 RedIRIS Autonomous System	42.8141	-1.6412
82	130.237.50.124	Sweden	2196583036	AS1653 SUNET Swedish University Network	59.3333	18.05
83	130.75.87.83	Germany	2185975635	AS680 service G-WiN	52.3667	9.7167
84	130.83.166.198	Germany	2186520262	AS8365 Man-da.de GmbH	49.8706	8.6494
85	130.83.166.199	Germany	2186520263	AS8365 Man-da.de GmbH	49.8706	8.6494
86	130.83.166.200	Germany	2186520264	AS8365 Man-da.de GmbH	49.8706	8.6494
87	130.92.70.252	Switzerland	2187085564	AS559 SWITCH, Swiss Education and Research Network	46.9167	7.4667
88	130.92.70.253	Switzerland	2187085565	AS559 SWITCH, Swiss Education and Research Network	46.9167	7.4667
89	131.130.32.153	Austria	2206343321	AS760 University of Vienna, Austria	48.2	16.3667
90	131.130.32.154	Austria	2206343322	AS760 University of Vienna, Austria	48.2	16.3667
91	131.175.17.10	Italy	2209288458	AS137 GARR Italian academic and research network	42.8333	12.8333

Figure (4.3) User Profile Matrix

Figure (4.4) shows the distribution of response time and throughput. Figure (4.4.a) shows that most of the response-time values are smaller than 1.6 seconds. Figure (4.4.b) shows that most throughput values are smaller than 64 kbps.



(a) Distribution of response time



(a) Distribution of throughput

Figure (4.4) QoS Values Distribution

A detailed statistical description for QoS data set of web services is presented in Table (4.1)

Table (4.1) Statistic of Web Service QoS Dataset

Statistics	Values
Num. of Records	1,974,675
Num. of Service Users	339
Num. of Web Services	5,825
Num. of User Countries	30
Num. of Web Service Countries	73
Scale of Response Time	0-20s
Mean of Response-Time	1.43 s
Standard Deviation of Response-Time	31.9 s
Scale of Throughput	0-1000kbps
Mean of Throughput	102.86 kbps
Standard Deviation of Throughput	531.85 kbps

4.3 Preprocessing Step

Initially, the original QoS dataset (response time and throughput) has been converted from text format to matrix format. Preprocessing step was taken place through the following processes:

- Deleting the services that have not been invocated by any user (empty records).
- Normalizing QoS values in order to divide the statistical interval more conveniently.

Table (4.2) illustrates a sample of the response time matrix after being pre-processed.

Table (4.2) Normalizing QoS Value of Response Time Matrix

	S_0	S_1	S_2	S_3	S_4	S_5		S_{5819}	S_{5820}	S_{5821}	S_{5822}	S_{5823}	S_{5824}
u_0	0.34	0.04	0.01	0.05	0.05	0.08	...	0.33	0.37	0.40	0.35	0.43	0.70
u_1	0.11	0.04	0.01	0.06	0.06	0.06	...	0.04	0.02	0.02	0.15	0.01	0.02
u_2	0.03	0.07	0.02	0.09	0.09	0.02	...	0.09	-1	-1	0.69	-1	-1
u_3	0.02	0.04	0.01	0.05	0.05	0.05	...	0.02	0.01	0.01	0.01	0.01	0.01
	...	...	...	...	...	...	...	...	...	...	...	...	...
u_{335}	0.02	0.04	0.01	0.05	0.05	0.06	...	0.02	0.01	0.01	0.01	0.01	0.01
u_{336}	0.05	0.06	0.02	0.11	0.08	0.10	...	0.11	0.04	0.05	0.04	0.06	0.04
u_{337}	0.05	0.13	0.03	0.98	0.08	0.08	...	0.08	0.03	0.04	0.03	0.08	0.06
u_{338}	0.05	0.03	0.01	0.05	0.05	0.07	...	0.02	0.01	0.01	0.01	0.01	0.01

4.4 User Reputation-based Calculation

The calculation of user reputation is described as follows:

First, we need to calculate the reliable interval of each service, then the user feedback information can be divided into trusted feedback, untrusted feedback and no feedback, according to the QoS value of the web service invoked by the user and reliable interval. After that, we count the feedback information of each user and express it as feedback vector. Finally, the reputation of users is calculated based on Dirichlet probability distribution.

In order to calculate reliable interval of each service, the QoS values in [0,1] intervals can be evenly divided into 10 intervals of (0,0.1], (0.1, 0.2], ..., (0.8,0.9], and (0.9, 1]. Trusted users always take the major part of all users. Table (4. 3) shows how to use statistical intervals to determine the reliable interval in service 0.

Table (4.3) Statistical Interval Results

Service 0

	0.0-0.09	0.1-0.19	0.2-0.29	0.3-0.39	0.4-0.49	0.5-0.59	0.6-0.69	0.7-0.79	0.8-0.89	0.9-0.1	No Invocation
u_0				0.34							
u_1		0.11									
u_2	0.03										
u_3	0.02										
u_4	0.03										
u_5	0.09										
u_6	0.02										
u_7	0.02										
u_8		0.1									
u_9	0.03										
u_{10}				0.36							
...	...	...	...	...	...	...	...	...	...	...	...
u_{81}											-1
...	...	...	...	...	...	...	...	...	...	...	...
u_{336}	0.05										
u_{337}	0.05										
u_{338}	0.05										

Table (4.3) presents an explanation of the results for QoS values of service 0 based on statistical interval. The first column (first interval) has contained most QoS values of users, and can be considered reliable interval for service 0. Table (4.4) shows whether users of service 0 are in the reliable interval, are not in the reliable interval, or have not invoked the service. The user's QoS value in the reliable interval can be considered as trusted feedback. Otherwise, it is untrusted feedback.

Table (4.4) User Feedback Information

Service 0

	If user in reliable intervale?	
u_0		No
u_1		No
u_2	Yes	
u_3	Yes	
u_4	Yes	
u_5	Yes	
u_6	Yes	
u_7	Yes	
u_8		No
u_9	Yes	
u_{10}		
...	...	...
u_{81}	No Invocation	
...	...	...
u_{336}	Yes	
u_{337}	Yes	
u_{338}	Yes	

After completing the calculation of all feedback information for all services, we count each user’s feedback information, and express the user’s feedback information as a feedback vector.

Table (4.5) shows the feedback vector for all users.

Table (4.5) User Feedback Vector

	User Feedback Vector		
	No Feedback	Untrusted Feedback	Trusted Feedback
u_0	193	2743	2889
u_1	275	572	4978
u_2	307	2005	3513
u_3	283	395	5147
u_4	267	384	5174
u_5	267	541	5017
u_6	273	440	5112
u_7	260	521	5044
u_8	270	412	5143
u_9	243	482	5100
u_{10}	193	3234	2398
...	...	...	...
u_{81}	193	3339	2293
...	...	...	...
u_{336}	300	1736	3789
u_{337}	304	1405	4116
u_{338}	294	624	4907

Finally, the user's reputation can be calculated from the feedback vector by using the probability expectation of the Dirichlet distribution. Table (4.6) shows that the identification of untrusted users depends on the threshold value (0.5). If the reputation value of the user is less than the threshold value, the user is identified as untrusted.

Table (4.6) Identifying Untrusted User

	User Reputation Value	Trusted/ Untrusted Depending on the threshold (0.5)
u_0	0.5129	Trusted
u_1	0.8967	Trusted
u_2	0.6365	Trusted
u_3	0.9285	Trusted
u_4	0.9307	Trusted
u_5	0.9025	Trusted
u_6	0.9205	Trusted
u_7	0.9062	Trusted
u_8	0.9256	Trusted
u_9	0.9135	Trusted
u_{10}	0.4258	Untrusted
...	...	...
u_{81}	0.4071	Untrusted
...	...	...
u_{336}	0.6857	Trusted
u_{337}	0.7454	Trusted
u_{338}	0.8870	Trusted

4.5 Clustering Users

One of the essential aspects that need to be taken into consideration are the users in the same geographical location tend to have similar QoS values for the same service, because they have a similar distance to the same provider and similar infrastructure and user that located in different location observed different QoS values for the same services. Thus, the DBSCAN clustering algorithm was used on the user side to form clusters according to latitude and longitude attributes of users with two input parameters, *minpts* = 2 and *Eps.* = 0.5. After implementing the DBSCAN algorithms, we formed 11 clusters.

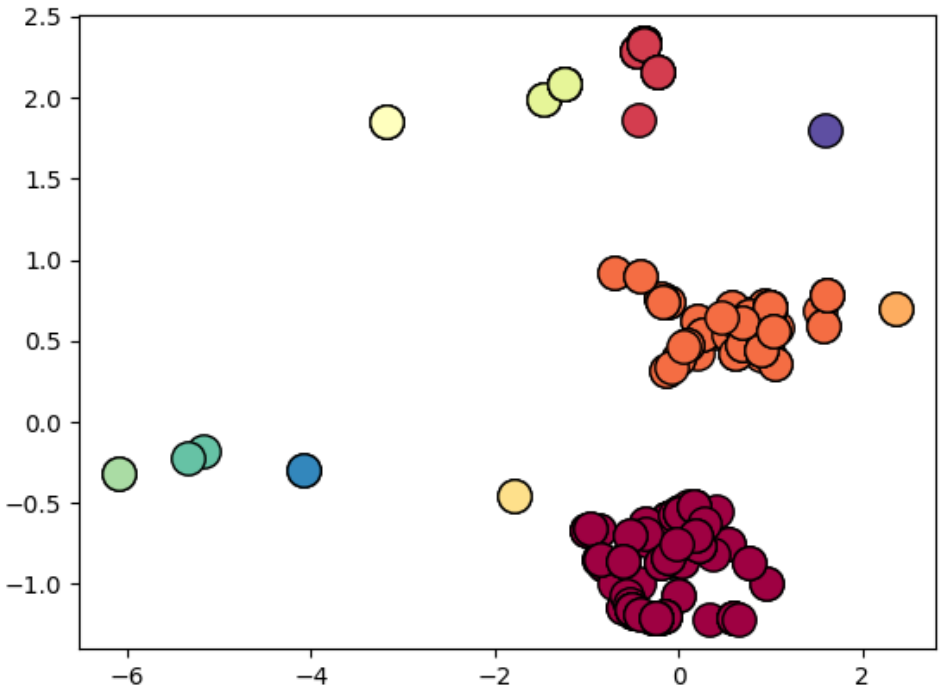


Figure (4.5) Clustering User Geographical Location

4.6 Predicting Unknow QoS values

Predicting Unknown QoS values depends essentially on the historical experience of neighboring users. Therefore, we cluster the users according to their location. After clustering the users, we predict the

unknown QoS values based on the values previously invoked by other neighboring users in the same cluster by computing the cosine similarity between users and then using the values of this similarity as weights.

To predict the invocation value of service S_0 for user u_0 in the table (4.7) that represent cluster 0, the cosine similarity computing between user u_0 and other users that have previously invocation values for S_0. As shown in the table (4.7), users (u1,u2,u4,u6,u8 and u9) have previously invocation for S_0, and the predict the missing value according to Equation (3.13).

Table (7.4) Cluster of QoS values

	Cluster 0									
	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
u_0		0.05	0.02		0.07	0.8		0.09	0.1	0.44
u_1	0.04	0.04		0.06	0.06	0.04	0.05		0.1	
u_2	0.02	0.05	0.01	0.06			0.05			0.35
u_3			0.01		0.05	0.06		0.05	0.07	0.03
u_4	0.04	0.04		0.05	0.05		0.03	0.05		
u_5		0.03	0.01			0.05	0.07		0.07	0.03
u_6	0.02		0.01	0.04	0.04			0.06	0.08	0.03
u_7		0.05			0.07	0.04	0.05	0.08	0.11	
u_8	0.04		0.02	0.07		0.05				0.04
u_9	0.17	0.05		0.06	0.06		0.06	0.09	0.1	0.05
u_{10}		0.05	0.01		0.05	0.15		0.06	0.07	

4.7 Performance Comparison

For demonstrating how accurate the proposed approach is in terms of predictions, the present section will draw a comparison between the results obtained through the proposed approach and those of some well-known QoS prediction approaches, including the following:

1. UPCC (User-based collaborative filtering method using Pearson Correlation Coefficient): The calculation of similarity among users is performed by means of PCC, and the prediction

of missing QoS values takes place according to the user's similarity [92].

2. IPCC (Item-based collaborative filtering method using Pearson Correlation Coefficient): As for the second method, the calculation of similarity between services is performed through PCC, and the prediction of missing QoS values occurs according to the service's similarity [93].
3. UIPCC: (User-Item-based collaborative filtering method using Pearson Correlation coefficient): This represents a hybrid approach whereby UPCC and IPCC are combined, and it utilizes the similarity of both users and services for improving the predicting accuracy [94] .
4. NMF (Nonnegative Matrix Factorization): It employs non-negative matrix factorization onto the user-service matrix for predicting missing QoS values [95].
5. Cloud-Pred (Cloud Prediction): It is a matrix factorization approach that uses both local and global usage information of both users and services when predicting [96].

In reality, the matrices of QoS attributes are very sparse. To evaluate the impact of matrix density on the prediction accuracy, some values are removed randomly from the matrices and the prediction results are compared with the results of other approaches. To exemplify, if the matrix density is 90%, this implies that 10% of the entries in the original matrix are randomly removed, leaving the remaining 90% as test data.

The experimental results of MAE and RMSE for various approaches at different densities are shown in Table (4.2) and Table (4.3), which in turn highlight the result of the proposed approach.

Table (4.2) Response-Time Performance Comparison Using MAE and RMSE Metrics.

Matrix density (%)	Metrics	Response-Time (seconds)					
		IPCC	UPCC	UIPCC	NMF	CloudPred	Proposed Approach
10	MAE	0.5173	0.5655	0.5654	0.6754	0.5306	**0.4710**
	RMSE	1.4874	1.3326	1.3309	1.5354	1.2904	**1.2693**
20	MAE	0.5135	0.5516	0.5053	0.6771	0.4745	**0.3506**
	RMSE	1.3419	1.3114	1.2486	1.5241	1.1973	**1.0942**
80	MAE	0.4785	0.4442	0.3873	0.3740	0.3704	**0.3159**
	RMSE	1.3414	1.1514	1.0785	1.1242	1.0597	**1.0031**
90	MAE	0.4605	0.4331	0.3793	0.3649	0.3638	**0.3031**
	RMSE	1.2761	1.1264	1.0592	1.1121	1.0359	**0.9570**
Average of MAE		0.4924	0.4986	0.4593	0.5228	0.4348	**0.3601**
Average of RMSE		1.3617	1.2304	1.1793	1.3239	1.1458	**1.0809**

Table (4.3) Throughput Performance Comparison Using MAE and RMSE Metrics.

Matrix density (%)	Metrics	Throughput (kbps)					
		IPCC	UPCC	UIPCC	NMF	CloudPred	Proposed Approach
10	MAE	27.019	26.201	22.656	19.770	19.000	**15.644**
	RMSE	63.002	61.965	57.465	57.376	51.823	**50.507**
20	MAE	26.195	21.931	18.123	15.779	15.420	**12.024**
	RMSE	60.399	56.544	50.043	50.140	44.897	**40.476**
80	MAE	25.558	14.549	12.488	12.510	10.788	**7.7303**
	RMSE	57.762	44.373	39.601	39.202	36.850	**34.335**
90	MAE	23.973	13.876	12.066	11.696	10.472	**7.5865**
	RMSE	54.882	42.553	38.076	36.755	35.922	**34.152**
Average of MAE		25.686	19.139	16.333	14.938	13.920	**10.746**
Average of RMSE		59.011	51.358	46.296	45.868	42.373	**39.867**

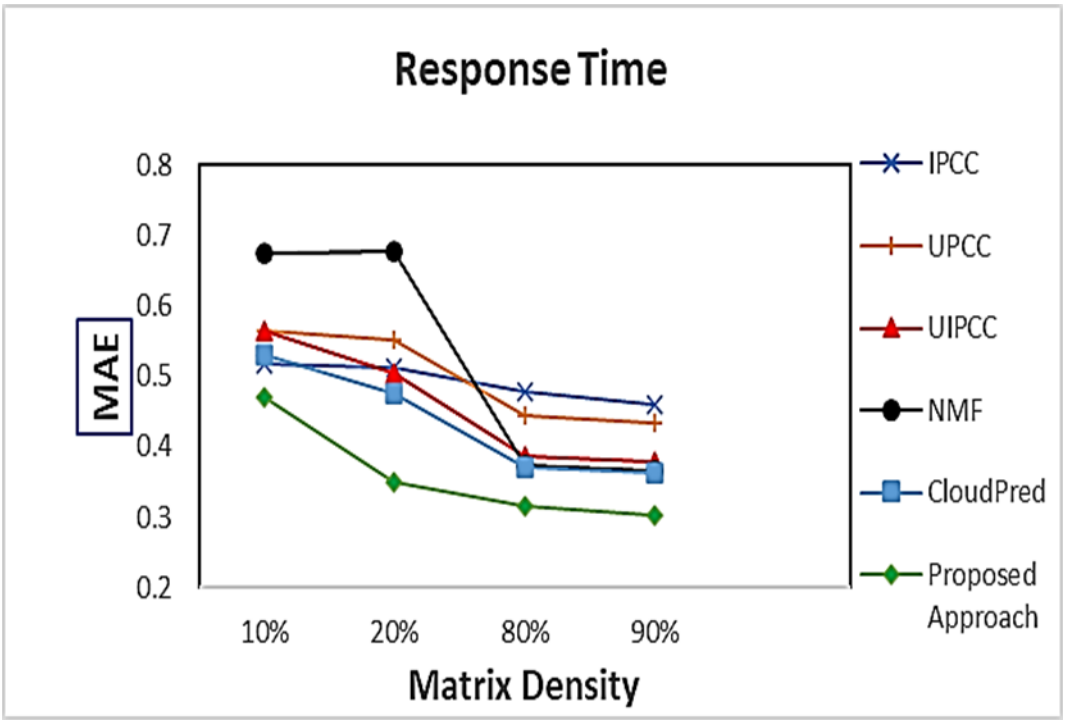
A number of significant aspects are observed in the experimental results:

1. At different matrix densities, it has been observed that that the proposed approach realizes relatively lower RMSE and MAE values than other methods for both response time and throughput attributes.
2. It has been noticed that the values of RMSE and NMA in sparse matrices (e.g., 10%; 20%) are larger than those of dense matrices (e.g., 90%; 80%), which refers to the fact that a sparser matrix provides less information for predicting unknown QoS values.
3. As compared to UIPCC, IPCC, UIPCC, NMF, and Cloud-Pred, the proposed approach obtained more accurate predictions. This can be traced back to the fact that the proposed method makes use of the geographical location of the users to compute the similarity of the neighbors. Thus, users sharing the same location tend to share similar QoS attribute values for the same service. Moreover, the proposed method uses reputation to identify dishonest users to guarantee data reliability by correcting unreliable data using QoS values for trusted users.

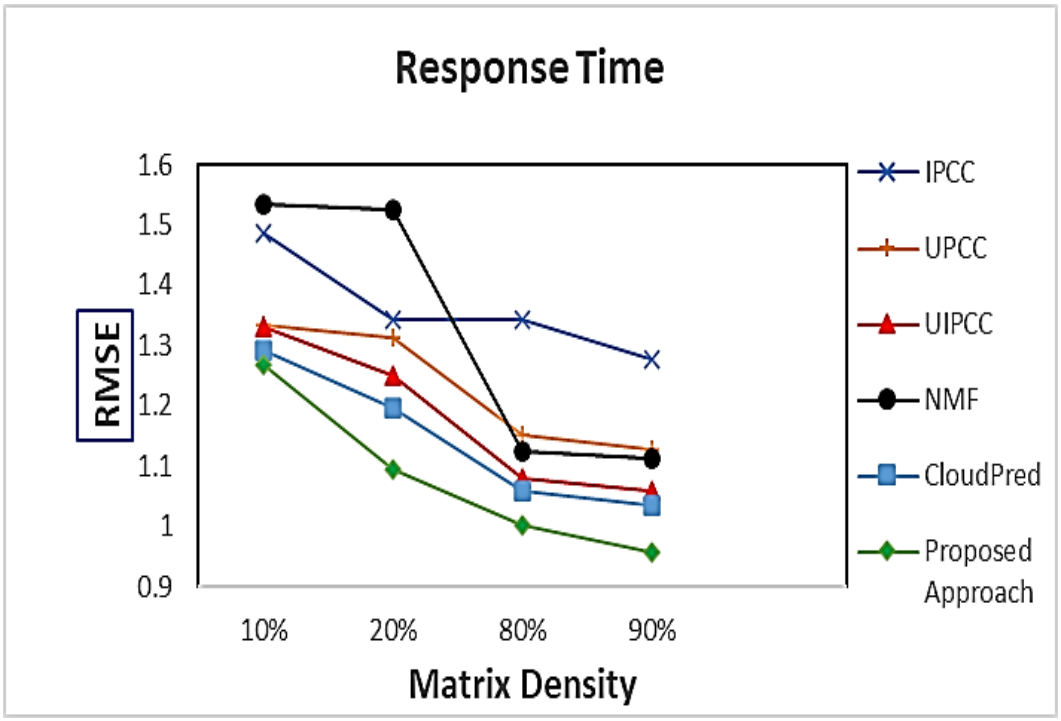
The values of RMSE and MAE of response time are lower than the throughput values, as the scale for the response time is 0-20s, whereas for throughput it is 0-100kbps

4.8 Impact of Matrix Density

The comparison of how accurate the predictions of different approaches at different matrix densities is shown in Figure (4.3) for response time attribute and Figure (4.4) for throughput attribute.

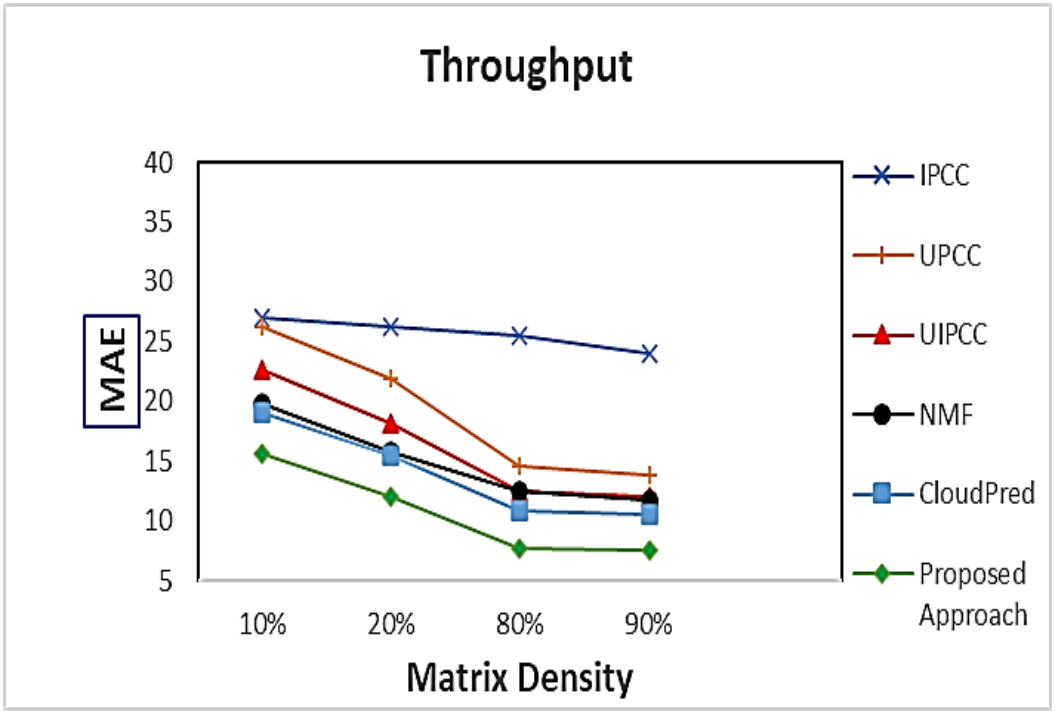


(a) (MAE)

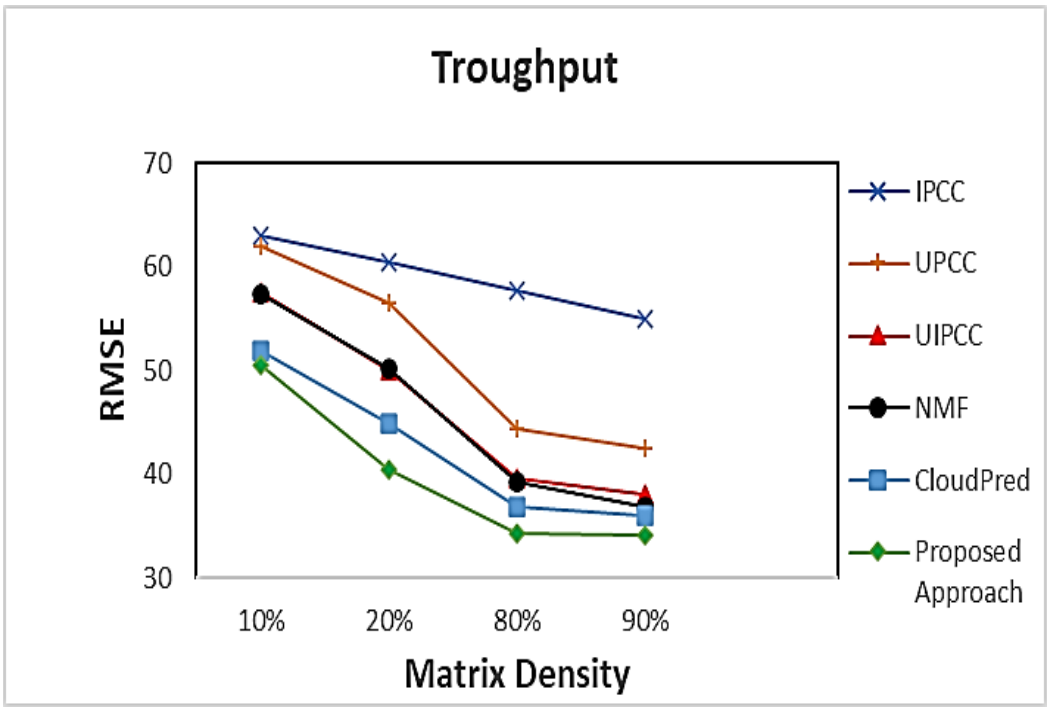


(b) (RMSE)

Figure (4.3) Impact of Response-Time Matrix Density



(a) (MAE)



(b) (RMSE)

Figure (4.4) Impact of Throughput Matrix Density

Figure (4.3) and Figure (4.4) show the accuracy of the prediction improves with the increase in matrix density. This indicates that the increase in matrix density makes more information available for prediction, leading to an increase in the accuracy of the prediction

Chapter Five

Conclusions and Future Works

5.1 Conclusions

Many of the QoS attributes of web services are dynamic and variable due to various factors such as users' reputation and network environment, and their values are different for each user. Therefore, personalized user-side QoS prediction is a necessity and essential to select the best services. This Dissertation,

proposes a new approach for personalized QoS prediction that combines user reputation and user geographic location based on collaborative filtering technique. It could be divided into three steps:

1. The user reputation is calculated using Dirichlet probability distribution for identifying dishonest users and then correct their unreliable data. The purpose of this step is for improving how accurate the QoS predictions are.
2. In order to have an effective similarity computation for users and also to alleviate the data sparsity problem, the users are clustered by their geographic location. As a result, the size of the computations is significantly reduced.
3. Finally, the missing QoS values are predicted in each cluster by weighting the similarity of neighboring users. This proved to be an effective method to improve prediction accuracy. It has been observed that users who share the same location tend to share QoS attribute values that are alike for the same services.
4. The experimental results indicate that the predictions are more accurate than those that are obtained by means of the traditional methods. As for the proposed approach, the missing QoS values are predicted based on the users' geographic location information only to determine the similarity of neighbors

5.2 Future Works

In this Dissertation, build a personalized QoS prediction based on Reputation and Geographic location. Where, the following are suggestions for future works:

1. Developing the system by combining users' information with services information as well as network environment information, where in this case may be enhancing performance with hybrid algorithm.
2. Designing a model for predicting unknown QoS values based on time-aware to improve the accuracy of prediction, since the response time attribute and the throughput attribute are strongly affected by the network environment using machine learning techniques to improve the prediction accuracy.

References

[1] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred : A Time-Aware Personalized QoS Prediction Framework for Web Services," *2011 22nd IEEE Int. Symp. Softw. Reliab. Eng. WSPred*, 2011, doi: 10.1109/ISSRE.2011.17.

[2] Y. Yang, Z. Zheng, X. Niu, and M. Tang, "A Location-Based Factorization Machine Model for Web Service QoS Prediction," *IEEE Trans. Serv. Comput.*, vol. PP, no. c, p. 1, 2018, doi: 10.1109/TSC.2018.2876532.

[3] Z. Chen, L. Shen, F. Li, D. You, and J. P. B. Mapetu, "Web service QoS prediction: when collaborative filtering meets data fluctuating in big-range," *World Wide Web*, vol. 23, no. 3, pp. 1715–1740, 2020, doi: 10.1007/s11280-020-00787-x.

[4] C. Wu, W. Qiu, Z. Zheng, X. Wang, and X. Yang, "QoS Prediction of Web Services based on Two-Phase K-Means Clustering," 2015, doi: 10.1109/ICWS.2015.31.

[5] S. Muthuraman, "Qualitative and Quantitative Review of QOS based Web Services Selection and Composition Techniques STUDY," 2016.

[6] L. Ren and W. Wang, "An SVM-based collaborative filtering approach for Top-N web services recommendation," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 531–543, 2018, doi: 10.1016/j.future.2017.07.027.

[7] R. S. N. and H. N. Nawaf, "QoS Prediction for Web Services Using User Based Classification," *J. Phys. Conf. Ser.*, vol. 1294, no. Issue 4, 2019.

[8] X. Luo, Y. Lv, R. Li, and Y. I. Chen, "Web Service QoS Prediction Based on Adaptive Dynamic Programming Using Fuzzy Neural Networks for Cloud Services," vol. 3, 2015.

[9] T. Chen, R. Bahsoon, and X. Yao, "Online QoS Modeling in the Cloud: A Hybrid and Adaptive Multi-Learners Approach," *2014 IEEE / ACM 7th Int. Conf. Util. Cloud Comput.*, 2014.

[10] M. R. Zhang, Y., Lyu, *QoS Prediction in Cloud and Service Computing Approaches and Applications*. SpringerBriefs in Computer Science, 2017.

[11] L. Kuang *et al.*, "A Personalized QoS Prediction Approach for {CPS} Service Recommendation Based on Reputation and Location-Aware Collaborative Filtering," *Sensors*, vol. 18, no. 5, p. 1556, 2018, doi: 10.3390/s18051556.

[12] S. Ran, "A model for web services discovery with QoS," *ACM SIGecom Exch.*, vol. 4, no. 1, pp. 1–10, 2003, doi: 10.1145/844357.844360.

[13] Y. Zhang, "Modeling and Exploiting QoS Prediction in Cloud and Service Computing," no. September, p. 190, 2013, [Online]. Available: http://search.proquest.com/docview/1547379235/abstract/5C88BAE8A14F49DCPQ/57?accountid=10218%5Cnfiles/2666/Zhang - 2013 - Modeling and Exploiting QoS Prediction in Cloud an.pdf%5Cnfiles/2759/57.html.

[14] S. R. Pat and P. V. P. Mapari, "A Survey On : Web Service Recommendation Using Location-Aware and Personalized Collaborative Filtering," vol. 6, no. 12, pp. 3625–3627, 2016.

[15] S. G. Deng, L. T. Huang, J. Wu, and Z. H. Wu, "Trust-based personalized service recommendation: A network perspective," *J. Comput. Sci. Technol.*, vol. 29, no. 1, pp. 69–80, 2014, doi: 10.1007/s11390-014-1412-2.

[16] S. Deng, L. Huang, and G. Xu, "Social network-based service

recommendation with trust enhancement," *Expert Syst. Appl.*, vol. 41, no. 18, pp. 8075–8084, 2014, doi: 10.1016/j.eswa.2014.07.012.

[17] Q. Xie, S. Zhao, Z. Zheng, J. Zhu, and M. R. Lyu, "Asymmetric correlation regularized matrix factorization for web service recommendation," in *2016 IEEE International Conference on Web Services (ICWS)*, 2016, pp. 204–211.

[18] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *IEEE Trans. ser-vices Comput.*, vol. 4, pp. 140–152, 2011.

[19] M. Tang, Y. Jiang, J. Liu, and X. Liu, "Location-aware collaborative filtering for QoS-based service recommendation," in *2012 IEEE 19th international conference on web services*, 2012, pp. 202–209.

[20] J. Xu, Z. Zheng, and M. R. Lyu, "Web Service Personalized Quality of Service Prediction via Reputation-Based Matrix Factorization," *IEEE Trans. Reliab.*, pp. 1–10, 2015.

[21] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "TAP: A personalized trust-aware QoS prediction approach for web service recommendation," *Knowledge-Based Syst.*, vol. 115, pp. 55–65, 2017.

[22] Z. Chen, L. Shen, and F. Li, "Exploiting Web service geographical neighborhood for collaborative QoS prediction," *Futur. Gener. Comput. Syst.*, vol. 68, pp. 248–259, 2017.

[23] X. Zhu *et al.*, "Similarity-maintaining Privacy Preservation and Location-aware Low-rank Matrix Factorization for QoS Prediction based Web Service Recommendation," *IEEE Trans. Serv. Comput.*, vol. 1374, no. c, pp. 1–14, 2018, doi: 10.1109/TSC.2018.2839741.

[24] L. Chen, F. Xie, Z. Zheng, and Y. Wu, "Predicting Quality of Service via Leveraging Location Information," *Complexity*, vol. 2019, 2019,

doi: 10.1155/2019/4932030.

[25] Z. Chen, Y. Sun, D. You, F. Li, and L. Shen, "An accurate and efficient web service QoS prediction model with wide-range awareness," *Futur. Gener. Comput. Syst.*, vol. 109, pp. 275–292, 2020.

[26] X. Wang, P. He, J. Zhang, and Z. Wang, "QoS Prediction of Web Services Based on Reputation-Aware Network Embedding," *IEEE Access*, vol. 8, pp. 161498–161508, 2020.

[27] M. Hasnain, M. F. Pasha, I. Ghani, M. Imran, M. Y. Alzahrani, and R. Budiarto, "Evaluating trust prediction and confusion matrix measures for web services ranking," *IEEE Access*, vol. 8, pp. 90847–90861, 2020.

[28] M. I. Smahi, F. Hadjila, C. Tibermacine, and A. Benamar, "A deep learning approach for collaborative prediction of Web service QoS," *Serv. Oriented Comput. Appl.*, vol. 15, no. 1, pp. 5–20, 2021.

[29] J. Yin and Y. Xu, "Personalised QoS-based Web service recommendation with service neighbourhood-enhanced matrix factorisation," *Int. J. Web Grid Serv.*, vol. 11, no. 1, pp. 39–56, 2015, doi: 10.1504/IJWGS.2015.067156.

[30] U. Javed, K. Shaukat, I. A. Hameed, F. Iqbal, T. M. Alam, and S. Luo, "A Review of Content-Based and Context-Based Recommendation Systems," *Int. J. Emerg. Technol. Learn.*, vol. 16, no. 3, pp. 274–306, 2021, doi: 10.3991/ijet.v16i03.18851.

[31] W. T. Tsai, X. Sun, and J. Balasooriya, "Service-oriented cloud computing architecture," *ITNG2010 - 7th Int. Conf. Inf. Technol. New Gener.*, pp. 684–689, 2010, doi: 10.1109/ITNG.2010.214.

[32] J. Zela Ruiz and C. M. Rubira, "Quality of Service Conflict during Web Service Monitoring: A Case Study," *Electron. Notes Theor.*

Comput. Sci., vol. 321, pp. 113–127, 2016, doi: 10.1016/j.entcs.2016.02.007.

[33] L. Rodero-Merino *et al.*, "From infrastructure delivery to service management in clouds," *Futur. Gener. Comput. Syst.*, vol. 26, no. 8, pp. 1226–1240, 2010, doi: 10.1016/j.future.2010.02.013.

[34] S. Murugesan and I. Bojanova, *Encyclopedia of cloud computing*. John Wiley & Sons, 2016.

[35] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "NIST cloud computing reference architecture," *Proc. - 2011 IEEE World Congr. Serv. Serv. 2011*, pp. 594–596, 2011, doi: 10.1109/SERVICES.2011.105.

[36] R. Buyya, J. Broberg, and A. Goscinski, *Cloud Computing: Principles and Paradigms*. 2011.

[37] B. Furht and A. Escalante, *Handbook of cloud computing*, vol. 3. Springer, 2010.

[38] K. Manasa Dhara, M. Dharmala, C. Krishan Sharma, K. Manasa, and C. Krishan, "A Survey Paper on Service Oriented Architecture Approach and Modern Web Services," *Surv. Pap. Serv. Oriented Archit. Approach Mod. Web Serv.*, p. 157, 2015, [Online]. Available: http://opus.govst.edu/capstoneshttp://opus.govst.edu/capstones/157.

[39] C. R. Choi and H. Y. Jeong, "A broker-based quality evaluation system for service selection according to the QoS preferences of users," *Inf. Sci. (Ny).*, vol. 277, pp. 553–566, 2014, doi: 10.1016/j.ins.2014.02.141.

[40] Z. Zheng, L. Xiaoli, M. Tang, F. Xie, and M. R. Lyu, "Web Service QoS Prediction via Collaborative Filtering: A Survey," *IEEE Trans. Serv. Comput.*, 2020.

[41] H. F. E. Yamany, M. A. M. Capretz, and D. S. Allison, "Quality of security service for web services with in SOA," *Serv. 2009 - 5th 2009 World Congr. Serv.*, no. PART 1, pp. 653–660, 2009, doi: 10.1109/SERVICES-I.2009.95.

[42] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," *Mob. Networks Appl.*, pp. 1–11, 2019.

[43] K. Kritikos, "Extending OWL for QoS-based web service description and discovery," *CEUR Workshop Proc.*, vol. 169, no. 4, pp. 73–78, 2005.

[44] E. M. Maximilien and M. P. Singh, "A framework and ontology for dynamic web, services selection," *IEEE Internet Comput.*, vol. 8, no. 5, pp. 84–93, 2004, doi: 10.1109/MIC.2004.27.

[45] A. K. Bardsiri and S. M. Hashemi, "QoS Metrics for Cloud Computing Services Evaluation," *Int. J. Intell. Syst. Appl.*, vol. 6, no. 12, pp. 27–33, 2014, doi: 10.5815/ijisa.2014.12.04.

[46] M. Oriol, J. Marco, and X. Franch, "Quality models for web services: A systematic mapping," *Inf. Softw. Technol.*, vol. 56, no. 10, pp. 1167–1182, 2014, doi: 10.1016/j.infsof.2014.03.012.

[47] Q. Tao, H. Y. Chang, C. Q. Gu, and Y. Yi, "A novel prediction approach for trustworthy QoS of web services," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3676–3681, 2012, doi: 10.1016/j.eswa.2011.09.060.

[48] Y. Ma, S. Wang, P. C. K. Hung, C. H. Hsu, Q. Sun, and F. Yang, "A highly accurate prediction algorithm for unknown web service QoS values," *IEEE Transactions on Services Computing*, vol. 9, no. 4. pp. 511–523, 2016, doi: 10.1109/TSC.2015.2407877.

[49] Y. Syu, J. Y. Kuo, and Y. Y. Fanjiang, "Time series forecasting for

dynamic quality of web services: An empirical study," *J. Syst. Softw.*, vol. 134, pp. 279–303, 2017, doi: 10.1016/j.jss.2017.09.011.

[50] Z. Zheng, H. Ma, M. R. Lyu, I. King, and H. Kong, "WSRec : A Collaborative Filtering Based Web Service Recommender System," *2009 IEEE Int. Conf. Web Serv.*, 2009, doi: 10.1109/ICWS.2009.30.

[51] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Trans. Serv. Comput.*, vol. 6, no. 3, pp. 289–299, 2012.

[52] J. Leskovec, A. Rajaraman, and J. D. Ullman, "Mining of Massive Datasets," *Cambridge University Press*. 2020.

[53] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Syst.*, vol. 46, pp. 109–132, 2013, doi: 10.1016/j.knosys.2013.03.012.

[54] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Adv. Artif. Intell.*, vol. 2009, no. Section 3, pp. 1–19, 2009, doi: 10.1155/2009/421425.

[55] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS Prediction for Web Services via Collaborative Filtering," *2007 IEEE Int. Conf. Web Serv.*, no. Icws, 2007.

[56] M. Tang, T. Zhang, J. Liu, and J. Chen, "Cloud service QoS prediction via exploiting collaborative filtering and location-based data smoothing," no. October, pp. 5826–5839, 2015, doi: 10.1002/cpe.

[57] A. Puri, M. Bhonsle, M. E. Student, and G. H. Raisoni, "A Survey of Web Service Recommendation Techniques based on QoS values," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 12. pp. 498–501, 2015, doi: 10.17148/IJARCCE.2015.412142.

[58] P. N. Shendage, “Review on Collaborative Filtering and Web Services Recommendation,” vol. 2, no. 6, pp. 912–916, 2014.

[59] J. Zhu, Y. Kang, Z. Zheng, and M. R. Lyu, “A clustering-based QoS prediction approach for web service recommendation,” *Proc. - 2012 15th IEEE Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput. Work. ISORCW 2012*, pp. 93–98, 2012, doi: 10.1109/ISORCW.2012.27.

[60] M. Fariss, N. El Allali, H. Asaidi, and M. Bellouki, “An Improved Approach for QoS Based Web Services Selection Using Clustering,” *Adv. Sci. Technol. Eng. Syst. J.*, vol. 6, no. 2, pp. 616–621, 2021, doi: 10.25046/aj060270.

[61] J. Liu, M. Tang, Z. Zheng, X. F. Liu, and S. Lyu, “Location-Aware and Personalized Collaborative Filtering for Web Service Recommendation,” *IEEE Trans. Serv. Comput.*, vol. 9, no. 5, pp. 686–699, 2016, doi: 10.1109/TSC.2015.2433251.

[62] S. Kumar, M. K. Pandey, A. Nath, K. Subbiah, and M. K. Singh, “Comparative study on machine learning techniques in predicting the QoS-values for web-services recommendations,” *Int. Conf. Comput. Commun. Autom. ICCCA 2015*, pp. 161–167, 2015, doi: 10.1109/CCAA.2015.7148398.

[63] H. Wu, Z. Zhang, J. Luo, K. Yue, and C. H. Hsu, “Multiple Attributes QoS Prediction via Deep Neural Model with Contexts,” *IEEE Trans. Serv. Comput.*, vol. PP, no. X, p. 1, 2018, doi: 10.1109/TSC.2018.2859986.

[64] G. White, A. Palade, C. Cabrera, and S. Clarke, “Autoencoders for QoS prediction at the Edge,” *2019 IEEE Int. Conf. Pervasive Comput. Commun. PerCom 2019*, pp. 1–9, 2019, doi:

10.1109/PERCOM.2019.8767397.

[65] K. Verbert *et al.*, "Context-aware recommender systems for learning: A survey and future challenges," *IEEE Trans. Learn. Technol.*, vol. 5, no. 4, pp. 318–335, 2012, doi: 10.1109/TLT.2012.11.

[66] A. R. Deshpande and M. Emmanuel, "Context based recommendation methods : a brief review," *Int. J. Comput. Appl.*, pp. 14–19, 2016.

[67] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone, "Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems," *RecSys'09 - Proc. 3rd ACM Conf. Recomm. Syst.*, pp. 265–268, 2009, doi: 10.1145/1639714.1639764.

[68] M. H. Zadeh and M. A. Seyyedi, "Qos Monitoring for Web Services by Time Series Forecasting," *Comput. Sci. Inf. Tech-nology*, 2010.

[69] W. Qiu, Z. Zheng, X. Wang, X. Yang, and M. R. Lyu, "Reputation-aware QoS value prediction of web services," in *2013 IEEE International Conference on Services Computing*, 2013, pp. 41–48.

[70] J. Yin, W. Lo, S. Deng, Y. Li, Z. Wu, and N. Xiong, "Colbar: A collaborative location-based regularization framework for QoS prediction," *Inf. Sci. (Ny).*, vol. 265, pp. 68–84, 2014, doi: 10.1016/j.ins.2013.12.007.

[71] J. Xu, Z. Zheng, and M. R. Lyu, "Web service personalized quality of service prediction via reputation-based matrix factorization," *IEEE Trans. Reliab.*, vol. 65, no. 1, pp. 28–37, 2015.

[72] S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, and F. Yang, "Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems," *IEEE Trans. Serv. Comput.*, vol. 8, no. 5, pp. 755–767, 2015, doi: 10.1109/TSC.2014.2320262.

[73] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol, *Recommender Systems Handbook*. 2011.

[74] N. Kumar, Y. S. Sneha, J. Mungara, and S. G. Raghavendra Prasad, "A Survey on Data Mining Methods Available for Recommendation System," *2nd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solut. CSITSS 2017*, pp. 278–283, 2018, doi: 10.1109/CSITSS.2017.8447672.

[75] G. Jain, T. Mahara, and K. N. Tripathi, "A Survey of Similarity Measures for Collaborative Filtering-Based Recommender System," *Adv. Intell. Syst. Comput.*, vol. 1053, no. January, pp. 343–352, 2020, doi: 10.1007/978-981-15-0751-9_32.

[76] J. Liu and Y. Chen, "A personalized clustering-based and reliable trust-aware QoS prediction approach for cloud service recommendation in cloud manufacturing," *Knowledge-Based Syst.*, vol. 174, pp. 43–56, 2019, doi: 10.1016/j.knosys.2019.02.032.

[77] D. Xu and Y. Tian, "A Comprehensive Survey of Clustering Algorithms," *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, 2015, doi: 10.1007/s40745-015-0040-1.

[78] S. Sirohi, N. Kumar, and A. Kumar, "a Detailed Study on Clustering Techniques and Tools for Data," *Int. Res. J. Eng. Technol.*, vol. Volume: 05, no. Issue: 04, p. 6, 2018.

[79] A. Fahim, "Homogeneous Densities Clustering Algorithm," *Int. J. Inf. Technol. Comput. Sci.*, vol. 10, no. 10, pp. 1–10, 2018, doi: 10.5815/ijitcs.2018.10.01.

[80] M. Daszykowski and B. Walczak, "Density-Based Clustering Methods," *Compr. Chemom.*, vol. 2, pp. 635–654, 2009, doi: 10.1016/B978-044452701-1.00067-3.

[81] T. Ali, S. Asghar, and N. A. Sajid, "Critical analysis of DBSCAN variations," *2010 Int. Conf. Inf. Emerg. Technol. ICIET 2010*, 2010, doi: 10.1109/ICIET.2010.5625720.

[82] D. Leung, *Denis Trček: Trust and reputation management systems: an e-business perspective*, vol. 20, no. 1–4. 2018.

[83] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, no. 2, pp. 618–644, 2007, doi: 10.1016/j.dss.2005.05.019.

[84] S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, and F. Yang, "Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems," *IEEE Trans. Serv. Comput.*, vol. 8, no. 5, pp. 755–767, 2015, doi: 10.1109/TSC.2014.2320262.

[85] P. Resnick and R. Zeckhauser, *Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System*, vol. 11. Elsevier Science, 2002.

[86] J. Schneider, G. Kortuem, J. Jager, S. Fickas, and Z. Segall, "Disseminating trust information in wearable communities," *Pers. Ubiquitous Comput.*, vol. 4, no. 4, pp. 245–248, 2000, doi: 10.1007/PL00000012.

[87] J. Lin, "On the dirichlet distribution," *Mater's Rep.*, 2016.

[88] A. Jøsang and J. Haller, "Dirichlet reputation systems," *Proc. - Second Int. Conf. Availability, Reliab. Secur. ARES 2007*, pp. 112–119, 2007, doi: 10.1109/ARES.2007.71.

[89] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed qos evaluation for real-world web services," in *2010 IEEE International Conference on Web Services*, 2010, pp. 83–90.

[90] M. M. Suarez-Alvarez, D. T. Pham, M. Y. Prostov, and Y. I. Prostov,

"Statistical approach to normalization of feature vectors and clustering of mixed datasets," *Proc. R. Soc. A Math. Phys. Eng. Sci.*, vol. 468, no. 2145, pp. 2630–2651, 2012, doi: 10.1098/rspa.2011.0704.

[91] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction forweb services via collaborative filtering," in *Ieee international conference on web services (icws 2007)*, 2007, pp. 439–446.

[92] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," *arXiv Prepr. arXiv1301.7363*, 2013.

[93] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[94] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 39–46.

[95] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[96] Y. Zhang and M. R. Lyu, *QoS Prediction in Cloud and Service Computing: Approaches and Applications*. Springer, 2017.

المستخلص

في السنوات الأخيرة ، أصبحت الحوسبة السحابية أكثر شعبية وقابلة للتطوير. تعتبر خدمة الوب المكون الاساسي لهذه التقنية. ومع ذلك، تلعب جودة الخدمة (QoS) لخدمات الوب دورًا مهمًا وأساسيا في اختيار أفضل الخدمات من عدد كبير من خدمات الويب ذات الوظائف المماثلة. لذلك، أصبح التقييم الفعال من جانب المستخدم لجودة خدمة الويب مجال بحثي مهم.

في هذه الاطروحة، تم إقتراح نهج هجين يجمع بين طريقة قائمة على السياق مع تقنية تصفية تعاونية (CF) للتنبؤ بقيم جودة الخدمة (QoS) لخدمات الويب واختيار أفضل الخدمات للمستخدمين النشطين باستخدام ما يلاحظه مستخدمين آخرين من جودة الخدمة للخدمات. ومع ذلك، في الطريقة المقترحة، تم استخدام اثنين من المعلومات السياقية الهامة: سمعة المستخدمين والموقع الجغرافي للمستخدمين. يتم استخدام السمعة لتحسين دقة التنبؤ، بينما يتم استخدام الموقع الجغرافي للمستخدين لتحسين حساب التشابه بين المستخدمين والتخفيف من مشكلة تبعثر البيانات.

تم تقسيم الأجزاء المقترحة من هذا العمل إلى ثلاث مراحل ، تتضمن المرحلة الأولى حساب درجة السمعة لكل مستخدم في مجموعة البيانات باستخدام توزيع احتمالية ديريشلي (Dirichlet) لتحديد المستخدمين غير الأمناء ثم تصحيح بياناتهم غير الموثوقة باستخدام معدل الفترة الموثوقة لجودة الخدمة في كل خدمة .المرحلة الثانية، يتم تجميع جميع المستخدمين حسب الموقع الجغرافي، حيث يمتلك المستخدمون الموجودون في نفس المجموعة قيم متقاربة من قيم جودة الخدمة .المرحلة الثالثة، مرحلة التنبؤ، وهي عملية التنبؤ بقيم جودة الخدمة غير المعروفة باستخدام نهج التصفية التعاوني المستند إلى التشابه بين المستخدمين في كل مجموعة.

كانت نتائج الاطروحة مرضية جدا مقارنة مع أعمال سابقة للتنبؤ بقيم جودة الخدمة غير المعروفة. أظهرت النتائج التجريبية أن الطريقة المقترحة تحسن بشكل كبير أداء التنبؤ بجودة الخدمة من حيث الكفاءة والدقة. كانت قياسات MAE و RMSE لسمة وقت الاستجابة، 0.3601 و 1.0809 على التوالي ، وقياسي MAE و RMSE لسمة الإنتاجية، 10.746 و 39.867 على التوالي.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل- كلية تكنولوجيا المعلومات
قسم البرمجيات

توقع جودة الخدمة للتوصية بخدمة الوب إستنادا على السمعة والتصفية التعاونية القائمة على عنقدة المواقع

مقدمة الى مجلس كلية تكنولوجيا المعلومات/جامعة بابل كجزء من متطلبات الحصول على درجة دكتوراه فلسفة في تكنولوجيا المعلومات/ برمجيات

مِن قِبَل

مؤيد نجم عبد الله ياسين

بأِشراف

أ.د. وسام سمير بهية

2021 A.D. **1443 A.H.**