DATA DEPENDENT WEB SERVICE SELECTION

by

Navati Jain

B.E. in Information Technology, PEC University of Technology

(Formerly Punjab Engineering College),

India, 2012

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the program of

Computer Science

Toronto, Ontario, Canada, 2015

AUTHOR’S DECLARATION

DATA DEPENDENT WEB SERVICE SELECTION

Navati Jain
Master of Science, Computer Science, 2015
Ryerson University

ABSTRACT

Data mining applications and services are becoming increasingly important, especially in this age of Big Data. QoS (Quality of Service) properties such as latency, reliability, response time of such services can vary based on the characteristics of the dataset being processed. The existing QoS-based web service selection methods are not adequate for ranking these types of services since they do not consider these dataset characteristics. We have proposed a service selection methodology to predict the QoS values for data analytic services based on the attributes of the dataset involved by incorporating a meta-learning approach. Subsequently we rank the services according to the predicted QoS values. The outcome of our experiments proves the effectiveness of this approach with an improvement of above 20% in service ranking when compared to the traditional QoS selection approach.

ACKNOWLEDGEMENTS

I am most thankful to my supervisor, Dr. Ding, who has patiently steered me through each stage of my thesis. Her keen interest and motivation has made this thesis possible and working under her guidance has truly been a rewarding experience for me.

I would like to thank the Department of Computer Science, my professors and classmates for their cooperation and assistance. I am also thankful to Mr. Pinder for his prompt responses to my queries and his proactive assistance during these graduate years.

I am very grateful to Dr. Ferworn, Dr. Harley and Dr. Woungang for taking the time to review my thesis and provide their insights and feedback.

My parents and little sister, Annanya, have been my biggest support during this critical period. I have been fortunate to meet some wonderful peers especially Poornima, Ssowjanya and Amar and I thank them for being my sounding board and giving me valuable advice.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AHP: Analytic Hierarchical Process

CSP: Constraint Satisfaction Problem

EM: Expectation Maximization

HTTP: Hypertext Transfer Protocol

k-NN: k Nearest Neighbor Algorithm

MAE: Mean Absolute Error

MCDM: Multi Criteria Decision Making

MLP: Multi Layer Perceptron

QoS: Quality of Service

REST: Representational State Transfer

SOA: Service-Oriented Architecture

SOAP: Simple Object Access Protocol

SOC: Service Oriented Computing

SRC: Spearman Rank Correlation

SVM: Support Vector Machine

UDDI: Universal Description Discovery and Integration

URI: Uniform Resource Identifier

WSDL: Web Service Description Language

XML: Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 Background and the Problem Statement

1.1.1 Background

In today's fast growing and demanding IT world, the technical needs of businesses rely on a wide variety of development languages, tools and platforms. Moreover, many organizations need to connect across the globe to enable inter- and intra-organization communication. Service-oriented computing (SOC) addresses many of these IT challenges by using services since services are platform independent, autonomous, loosely coupled as well as support rapid and low cost distributed application development [1]. Services are the fundamental elements that can be used for creating software applications that are made available across a network.

A web service is a service that can be used across the Internet and is identified by a URI (Uniform Resource Identifier). Web services can be implemented using XML (Extensible Markup Language) based interfaces such as WSDL (Web Service Description Language) [2]. The SOAP (Simple Object Access Protocol) specification is an XML based protocol used to access web services across the Internet. Recently, the REST (Representational State Transfer) protocol has become widely adopted for web service creation. The REST architectural style is based on HTTP (Hypertext Transfer Protocol) standard methods used to retrieve or manipulate resources.

Service Oriented Architecture (SOA) is an architectural model that can be used to logically support service oriented computing [2]. It typically comprises of three entities: the service provider, the service consumer and the service registry. Service providers can publish their services, their constraints and descriptions in the service registry. Service consumers can find and select such services through the registry. The UDDI (Universal Description Discovery and Integration) is an open specification of a registry where services can be discovered using approaches such as keyword based search engines and category browsing [3].

SOA can also include a service broker as a trusted third party to ensure that service providers comply with privacy and security regulations. It can assist the service requestor to find a provider from a list of service providers it manages based on the requirements.

The implementation backbone for an SOA is the Enterprise Service Bus (ESB). The ESB is a message bus that supports interoperability among different service applications. It uses message protocols to ensure an efficient control, flow and translation of messages between services [4].

As more services become available, there is a need to be able to identify the most appropriate web service for a specific application. Various efforts have been made to utilize syntactic, semantic and structural information of web service specifications as well as build extensive service description and publication techniques [5]. Web services can be described on the basis of their functional and non-functional aspects. The functional value of a web service is used to describe what a web service does and the non-functional values describe how the web service supports what it offers [3]. To distinguish among functionally similar web services, non-functional attributes or Quality of Service (QoS) attributes can be considered. For instance, to differentiate among several hotel booking reservation services that are functionally similar to each other, a user can provide his QoS attribute constraints such as the “cost of using the web service” should be less than $5, the “availability of service” should be greater than 90% and the “reliability of the web service” should be greater than 80% in a high peak vacation time to find services that best satisfy his needs.

Service matchmaking refers to the process of identifying services that meet the specified requirements of a service client. This process is used to determine which services meet the requestor’s QoS needs. Service matchmaking is used to filter services from the list of available services. Service ranking sorts the matched services based on the degree to which the service meets the specified requirements.

1.1.2 Problem Statement

Various approaches for QoS based web service selection have been proposed in recent literature. There are vector and matrix based approaches [6-8] in which provided QoS values and QoS requirements are represented in a vector form. The services are then ranked based on the distance between the vector representing the available QoS values and the request vector. Utility based approaches in [9, 10] optimize utility functions that are used to represent the QoS requirements. The MCDM (Multi Criteria Decision Making) approach in [11-14] is used to prioritize among different types of QoS (such as required QoS attributes versus optional QoS attributes) during service selection.

We consider a situation where a web service is used to process data, for example data analytic web services such as classification or clustering services. In a world which is leaning towards big data and data analytics, data mining algorithms and predictive analytics have become more applicable and businesses are taking initiative and interest in investing into applying such techniques. However, running such algorithms on a client machine may not be an ideal solution considering the volume of data, the speed of the incoming data and the cost of using such software. Hence, a client would prefer to use data analytic web services to process his data. In such a scenario, the aforementioned web service selection methods would not suffice in recommending web services. This is because some of the QoS values for services would vary for different datasets. For example, a large dataset is expected to take longer for such services to process and can thus affect QoS performance attributes such as “response time” and “latency”. The “reliability” of a service can be negatively affected if the algorithm that the service implements is sensitive to certain meta-attributes. For example, a service that cannot handle missing data at all will fail on a dataset with missing values, which will thus decrease its reliability.

The existing service selection schemes need to take into consideration the impact of dataset being processed on the QoS values of a service to support the recommendation of web services most suitable for the dataset. In other words, the service selection algorithm needs to incorporate “data-dependent QoS attributes”. We can further categorize such “data-dependent QoS attributes” as “per dataset data-dependent QoS attributes” and “per service data-dependent QoS attributes”. The “per dataset data-dependent QoS attributes” can include attributes such as latency, accuracy and response time as these attributes change for a service based on the specific dataset involved. The “per service data-dependent QoS attributes” on the other hand can include measures such as reliability and throughput since such metrics measure the overall quality of the service based on all the datasets they have dealt with. However, not all QoS values are data dependent, some QoS measures such as availability, interoperability, integrity are not affected by the data involved.

Our focus is on services that process data such as data analytic services, and the problem we would like to solve in this work is to come up with a selection approach which considers not only a service’s QoS properties, but also the meta-attributes (e.g., dataset size, data type, number of dimensions, etc.) of the dataset to be processed, and then rank or recommend data analytic services based on how well they deal with the given dataset.

1.2 Objectives

In this thesis, we have proposed a QoS service selection system for data analytic services that incorporates the data characteristics i.e. the meta-attributes of the dataset involved. For our work, we have considered data clustering services as an example of such data analytic services. The previous methods for QoS based selection may not suffice in recommendation of such data analytic services as the QoS values of the service may vary based on the kind of the dataset being processed. To facilitate the selection of data analytic services, we have proposed to use a meta-learning algorithm that predicts QoS values of services for a particular dataset, and then performs selection based on the predicted values.

Our objectives include the following:

1) To propose an algorithm that can be used to predict QoS values of data analytic web services for a new dataset to be processed;
2) To provide a recommendation and ranking mechanism of data analytic services based on their predicted QoS values.

1.3 Methodology

Our approach towards recommending data analytic services involves training the framework with different datasets. Every time the system is used for running a service on a dataset, we can collect the information describing the datasets, i.e. “meta-attributes” as well as the QoS data based on the service performance. This collected data can be used to maintain a log of QoS values that can vary based on the kind of dataset. For the purpose of our experiments, we generated many synthetic datasets and applied the services on them. This collected data can be used as the data-repository for the system.

To rank services for a new dataset, we can identify datasets similar to it from the existing dataset repository. After having discovered similar datasets, we can predict the “per dataset data-dependent QoS attributes”. In our thesis, we used accuracy and latency as “per dataset data-dependent QoS attributes”. We also update the “per service data-dependent QoS attributes” such as reliability to reflect the overall behavior of the service based on the service performance for all the datasets used by the service.

The estimated QoS values are in turn used as input to the QoS based service selection module to rank the services for the new dataset. In our framework, we have used a utility based selection technique to rank the services based on QoS values. This way, a service consumer can make a better decision about which service to select without having to try all the services.

1.4 Thesis Outline

The rest of the thesis is organized as follows:

In Chapter 2, we discuss some of the clustering methods we used in our services. We present a review that spans over various QoS driven service selection techniques proposed in the past. We also review some meta-learning methods that have been used to predict the ranking of machine learning algorithms.

In Chapter 3, the methodology of our system is presented. We explain how we calculated different QoS values for the services we have considered. We describe our similarity and prediction calculation approach. Finally, we provide a description of the service selection technique used in our framework.

In Chapter 4, we discuss our experiment design and the tools and system configuration. We discuss the data generator for creating the synthetic datasets used in our experiments. We also describe the metrics used for evaluating our approach. We present our experimental results and discuss our observations.

Finally, we conclude our thesis in Chapter 5 with a summary of our methodology and results. We have also mentioned some suggestions for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

There are two types of related work we would like to discuss in this chapter: i) QoS Based Web Service Selection and ii) Meta learning approaches. Before we go on to review these related works, we would first like to discuss the clustering algorithms that have been chosen as the representative data analytic algorithms for implementation of our services in the later experiment. We then present a review for various QoS based web service selection methodologies. We also discuss some meta-learning techniques that can be applied to provide recommendation of machine learning algorithms.

2.2 Clustering Methods

Clustering is a process which divides a set of data points into groups in such a way that the data points within a group are similar to each other while they remain dissimilar to points in other groups. Data clustering can be applied for a wide span of problems in image analysis, web search, social media, information retrieval, marketing etc. For example, in the area of image analysis, clustering can be used in handwritten character recognition systems based on the lexemes used in the text [15].

The k-means algorithm is a classic clustering algorithm which begins by assigning ‘k’ random data objects as the center of ‘k’ clusters. It then places the rest of the data points in these ‘k’ clusters according to a similarity measure such as the Euclidean distance between the center and the data point. Next, the algorithm iteratively tries to improve the within-cluster variation by recalculating the cluster centers based on the current memberships and subsequently reassigns the data objects according to the updated centers. The iteration continues until the assignment of the data points in their respective clusters is stable [16].

Several variations to the k-means algorithms have been proposed. The x-means [17] algorithm extends the k-means algorithm using the 'Improve Structure' operation to determine which subset of the current centroids needs to be split. This information is used to decide where the new centroids should be placed. The set of centroids with the best 'Bayesian Information Criterion' score is used as the final output.

The EM (Expectation Maximization) clustering algorithm, another k-means variant, employs a probabilistic approach in which each iteration comprises of two main steps : the 'expectation' step in which cluster probabilities are computed and the 'maximization' step that attempts to maximize the similarity of the objects in a cluster by adjusting the center of the cluster [16, 18] . The Farthest First algorithm based on the farthest first traversal algorithm in [19] is another version of the k-means algorithm and is comparatively simpler and faster [18].

A global metric based on 'cluster histograms' which is generated based on size of a cluster is used in the CLOPE algorithm to cluster categorical data. A parameter called repulsion is used to control the extent of similarity of the objects within a cluster [20].

The COBWEB clustering method is a clustering technique that produces a hierarchical tree, the nodes of which represent a cluster and the probability distribution associated with it. It supports incremental clustering and accommodates a new object in the tree based on the 'category utility', an evaluation metric used to maximize intra cluster similarity and inter cluster dissimilarity [21, 22].

The notion of density based clustering is used in the DBSCAN algorithm which identifies clusters based on the neighborhood of the points in a dataset. The specified minimum number of points required to form a cluster and the distance between the points is used to define the neighborhood of a point. Data points that do not satisfy these cluster requirements are considered as outliers [23].

Hierarchical clustering methods organize data points in a hierarchical manner to determine groups of clusters and their sub groups. There are two main approaches to achieve hierarchical clustering, the agglomerative method (the bottom up approach) and the divisive method (the top down approach). The agglomerative method initially treats each data point as a cluster and merges the clusters in each step to create bigger clusters. The divisive method on the other hand, begins by considering one cluster comprising of all the data points which is split into smaller clusters iteratively [16].

Various agglomerative clustering techniques are based on the criteria used to determine similarity between clusters based on the linkage measure. The single linkage algorithm is based on the link between the nearest clusters and the shortest distance between the clusters controls the merge of two clusters. Conversely, the complete linkage algorithm allows clusters to fuse according to the maximum distance between the closest clusters [16].

The average link clustering technique utilizes the average distance between every pair of member instances of the two clusters. The Ward clustering link type method aims to minimize the increase in the distance between the clusters prior to merging them in iteration. The distance used for Ward clustering is the sum of the squares of the distance of the data points from the centroid [18].

2.3 QoS Based Web Service Selection

Web services are software resources that are well defined and self-contained, can be assembled to provide business functionalities and can be advertised across the Internet [4]. Web services are able to support various processes as well as the creation of dynamic web sites. Since many more web services are being developed by various providers, we need to be able to choose services that best meet the requirements at hand. To differentiate among services in a pool of functionally equivalent services, the non-functional attributes or QoS (Quality of Services) attributes such as response time, latency, reliability, availability, accessibility, etc. can be evaluated to distinguish services based on such quality criteria.

The QoS attributes can include several measures [24] [7] that can be used to assess the performance of a service. Reliability indicates how many error messages have been generated compared to the total number of messages. Latency is considered as the time taken for the request to be processed. Response Time is the time it takes to send a service request and receive the response. Throughput is another performance measure that represents the maximum number of invocations that can be managed in a certain time period. Availability is defined as the percentage of time the service is available. Accessibility can be used to assess if the system is functioning normally or if the requests can be handled without delay.

The QoS based service selection process can entail both service matching and service ranking. Service matching is done to match a specified request with available services to choose among several services. The matched services can then be ranked based on the degree to which they fulfill the requirements.

Preferences for specific quality measures can be incorporated in the selection process by assigning weights to the QoS metric. Demands including user demands on different QoS metrics can be expressed as constraints. To optimize the service selection process, several approaches such as constraint programming (CP), mixed integer programming, multi-criteria decision making approaches (MDCM) have been applied in recent works as discussed further on.

QoS values for each web service are represented in two-dimensional matrix form in [6-8]. These values are then normalized to ensure a uniform measurement independent of the units used. In the model proposed in [6], the quality matrix undergoes an additional phase of normalization to allow the representation and manipulation of groups of quality criterion. The matrix representation facilitates service ranking in [7] by incorporating user specified weights and aggregating the values in each row, where each row represents a web service. Yan and Piao [8] discuss a matching algorithm which selects the services that meet the client requests based on the compulsory constraints and the relationship statements specified. The matched web services are then ranked by computing the distance between the vector representing the requirements and the vector representing the published QoS attributes.

In [9], a global utility function is built as a weighted function of individual utility functions where each individual utility function is used for different QoS metrics such as response time and throughput. This function is used by a service broker to select a service provider by maximizing the user-provided utility function under the cost constraints. The framework here employs a predictive analytic performance model to estimate the values of such metrics for the current workload commitments of each provider. Another utility based approach is used in [10]. This method uses a declarative logic based mechanism to provide service selection. The ranking is based on optimizing utility policies using optimization techniques including linear programming.

The model in [25] allows two-way service matchmaking by checking conformance of the provider's offer to the client's demand after ensuring that both the constraints in the demand and offer are consistent with each other. A constraint can be represented as a relation among the different variables. The offers are then optimized based on weighted combination of utility functions by considering it to be a constraint satisfaction problem (CSP). A CSP is a problem that is expressed in the form of constraints.

The QoS-based web service matchmaking task in [26] is done using an MIP (Mixed Integer Programming) engine for linear constraints and CSP engine similar to [25] for non-linear constraints. The framework here also provides an advanced categorization of the results: super offers, exact offers, partial offers and failed offers based on how well the services have been matched to the constraints in the demand.

The semantic QoS aware framework in [27] aims to find services that are semantically compatible in terms of their QoS attributes. The Semantic Matchmaking is done by applying description logic reasoning. The QoS conditions are translated to constraints and then checked for conformance using Constraint Programming. Finally, a vector based ranking approach that considers the tendency and weights associated with the QoS attribute is used to sort the candidate services.

Multi Criteria Decision Making (MCDM) methods can be implemented to assist in service selection based on multiple priorities. Such techniques have been used to incorporate tradeoffs between QoS characteristics [11, 12]. Analytic Hierarchical Process (AHP) is a popular technique that solves problems by modeling them in the form of a hierarchy. This hierarchy is created by dividing the multiple criteria involved into sub-criteria and further dividing them again into sub-criteria and thus establishing different levels of criteria with the solution alternatives in the final level. The priorities associated with the criteria and their sub criteria is incorporated to assist in computing the relative ranking of the various options within each level and finally determine the overall rank of each alternative [12].

The QoS-based ranking procedure in [12] implements the AHP method along with a supporting ontology. This approach entails four phases. First, the AHP hierarchy is generated to assist in establishing the importance levels between a pair of QoS related criteria such as user constraints versus system constraints, required constraints versus optional constraints etc. In this process, groups and sub-groups of QoS criteria are created at each level of the hierarchy. The candidate services form the alternative solutions of the AHP hierarchy. The normalized weight vector is computed for each group. The “service relative ranking” matrix is then calculated by considering predefined QoS comparison rules. Ultimately, the final ranking vector is derived from the “service relative rank” vectors of the QoS criteria and their associated weights. This vector is then sorted to acquire the ranked list of services.

Another MCDM approach presented in [11] is a generic one that can be integrated with existing selection techniques that have not accommodated priorities. This methodology is an extension of the “UML QoS Framework” to allow representation of priorities by introducing new meta-classes. The “PROMETHEE method” is adopted here as a class of outranking methods to compare the alternatives in a pairwise fashion and generate a ranking.

Skyline based approaches have also been employed as another method towards multi criteria matching of web services. A skyline is the subset of points in a d-dimensional space that are not dominated by any other point. This concept of “dominance” has been applied in the service selection proposal in [13]. For a particular object, the “dominated score” specifies the average number of objects that dominate it, while the “dominating score” represents average number of objects that it dominates. “Dominance score” takes both of these measures into consideration with the help of a scaling factor. Three algorithms for ranking web services based on these three scores are presented in this research with the aim of retrieving the top specified number of dominant web services and combining the degrees of match for various parameters.

The methodology in [14] takes into consideration the issue of uncertain QoS values and the notion of “p-dominant service skyline” to perform service optimization where ‘p’ is the probability of a service provider being dominated by another provider. A two-phase algorithm which makes use of a p-R-tree indexing structure to calculate the p-dominant service skyline is discussed. The p-dominated providers are pruned out using a dual-pruning scheme and then the dominant probability is calculated on the rest of the providers.

Fuzzy logic theory concepts can be applied to perform service selection as well. A personalized web service selection UDDI based architecture is discussed in [28] which takes into consideration, both the objective information given by the service providers as well as the subjective information supplied by users. This system architecture enables the interaction between the client, the QoS Agent and the service provider. The client is responsible for handling user requests and responses while the service provider can provide QoS information to the extended UDDI registry. The QoS agent contains modules to handle client’s QoS requests, to monitor the QoS information and evaluate it to make recommendations. The fuzzy expert component uses the objective information to provide the final QoS matching degree after two stages of fuzzy inference. A genetic algorithm is used to modify the objective information to enhance the accuracy of the fuzzy inference. Finally, the average recommended information is obtained by calculating the “trustability” of the users and similarity between them. The recommender system generates the list of the top web services in accordance to user preferences.

We have reviewed the existing service selection approaches that aim to create a mechanism to assist in choosing among functionally equivalent services based on their QoS aspect. To the best of our knowledge, no paper has provided a QoS selection mechanism specific for data analytic services or considering the effects of data attributes on QoS values. In this thesis, we have discussed an approach to facilitate selection for data-analytic services considering that the quality of services can vary for different types of data.

There have been some recent works [29-31], that address requirements and models to support the implementation of data mining services. The proposed model in [29] has two main components for implementing a data mining service: the “mining data” which is the data source and the “mining engine” which serves as the main building block and is responsible for mining tasks such as building the model, training it and evaluating it. Both models in [29, 30] support QoS negotiation though they have not discussed an approach towards QoS service selection. The K-Grid in [31] is a framework that allows the execution of data mining applications as services on the Grid.

2.4 Meta Learning Approaches

Algorithm selection in machine learning is a challenging task as it often relies on expensive trial and error techniques and specialized knowledge of the analyst. The meta-learning approach can assist in the recommendation of data mining algorithms based on previous performance results of these algorithms. Meta-learning techniques can be used to study the relationship between characteristics of the problem such as the characteristics of the dataset and performance of the machine learning algorithm.

Meta-attributes can be used to measure the dataset characteristics. The meta-attributes can include simple measures that can be extracted directly (e.g., the number of examples in a dataset), to statistical measures that are based on the probability distribution of the dataset (e.g., the skewness of the attributes), to information theoretic metrics that measure the entropy of attributes (e.g., mutual information) [32]. In many of the meta-learning frameworks, the performance of the candidate algorithms is assessed based on a single measure such as the accuracy of the algorithm. The learning process in the meta-learning system can be executed by machine learning algorithm such as decision trees and neural networks [33]. The knowledge acquired in this process can be used to recommend machine learning algorithms for a new dataset.

The main application area towards selection of machine learning algorithms has been in the recommendation of classification algorithms [34]. In [35], the authors have attempted to generalize the learning approach for selection of algorithms to other domains such as regression, time series forecasting, sorting, constraint satisfaction problems and optimization. There has also been some recent research on applying meta-learning techniques to predict the performance of clustering algorithms such as [33, 36, 37] based on a set of meta attributes describing the dataset.

The meta-learning approach in [33, 36, 37] employs a meta-learning algorithm to learn the relation between the set of meta-attributes and the ranking of the algorithms. The system is initially trained with a set of datasets based on the extracted meta-attributes that characterize these datasets. The clustering algorithms are ranked for each dataset based on a specific evaluation metric that assesses the quality of the clustering results such as the ‘global error rate’ in [36] or FBCubed Metric in [33]. In this manner, a recommendation can be made for a new dataset according to its meta-attributes.

In [33], ranks of some clustering algorithms on selected UCI datasets [38] were established based on the calculated FBCubed Measure. The k Nearest Neighbor algorithm (kNN), Multi-

Layer Perceptron (MLP), neural network, the Decision Tree and Naive Bayes were used as the meta learning algorithms to map the relationship between the proposed meta attributes and the ranking. Their findings showed that the kNN algorithm performed the best for their datasets.

The Support Vector Machine (SVM) regression algorithm is used as the meta-learner in [37]. The SVM algorithm was chosen by the authors based on their preliminary analysis that indicated that SVM provided better accuracy than neural networks and kNN. The SVM based meta-learner was applied on cancer gene expression microarray datasets to suggest ranking of different clustering algorithms including the k-means algorithm, Single Linkage, Complete Linkage, Average Linkage, Spectral clustering etc.

The above framework is extended in [36] with a different set of meta-attributes and the algorithms were ranked using Multilayer Perceptron network and Support Vector Regression as the meta-learners. The experiments here are based on artificially generated datasets. Several combinations of different dataset characteristics and cluster structures were used to train the system to work with different types of datasets.

Brazdil et al [39] use a different approach towards recommendation of classification algorithms. The presented model here first identifies similar datasets to the query dataset based on the expectation that the performance of a classification algorithm should be comparable for similar datasets. Using this knowledge, a ranking can be generated based on the performance of the similar datasets. To identify these similar datasets, the kNN algorithm has been applied. Unlike [33, 36, 37], this approach considers multiple criteria for ranking of classification algorithms. The ranking of algorithms incorporates the speed of the classification algorithms in addition to their accuracy using a method called "Adjusted Ratio of Rank (ARR)" in which the advantage of one algorithm over the other is represented as a ratio.

Vukićević et al [40] opine that a cloud-based architecture would enable an efficient meta-learning system since the size of the problem and algorithm space can affect the accuracy of the meta-model. They proposed an extended meta-learning system that clusters biomedical data such as gene expression and cell type classification datasets using a component based design. Using this model, a hybrid algorithm is created according to the meta-attributes of the current problem. The hybrid algorithm is developed by combining components that are reusable among similar structured algorithms that have common internal functionalities such as the distance measure used and cluster evaluation metrics.

The meta-learning technique in [41] uses the "RIPPER algorithm" as a rule learner to generate rules for a pair of algorithms which can describe which algorithm performs better than the other. These pair wise meta-rules are then converted to meta-attributes. The "Approximate Ranking Tree Forest" is used to predict the ranking of supervised algorithms.

Since meta-learning techniques provide recommendation based on the previous datasets used to train the learner, an experiment database can be used to ensure an effective implementation of meta-learning systems. Experiment databases store information associated with experiments which includes the datasets used, the algorithms and corresponding parameter settings applied along with the results [42]. An experimental database has been implemented in [42], and the authors have extended it in various works including supporting ontologies and a description language in [43].

An experimental database can provide a wealth of knowledge to analyze and enhance research experiments. The adoption of such a system can ensure the reproducibility and reusability of results and hence save researchers the time and effort of having to perform previously conducted experiments for benchmarking and comparison purposes [43]. Their application can thus be extended to meta-learning systems as the information pertaining to the meta-attributes, the performance features of the candidate algorithms, the evaluation criteria and results used can be collected in an organized manner and assist in refining the meta-learning system.

Our approach differs from the methodology used in these meta-learning algorithms. We have also used a meta-learning approach in our work to recommend data analytic services for different datasets. Our work focuses on multiple QoS attribute selection unlike the selection of algorithms based on a single performance measure approach used in [33, 36, 37]. The work in [39] is also based on a multi-criteria evaluation approach towards selection of classification algorithms. In their work, the similar datasets are used to estimate the rank of an algorithm where the ranking combines success rate and time. Our work uses a set of identified similar datasets to predict the individual QoS attribute values (i.e. the "per dataset data-dependent QoS attribute" values). We have also included "per service data-dependent QoS attributes" that can be estimated based on the service's behavior on past datasets. These estimated values can then be used to rank the service. One advantage of predicting QoS values instead of only predicting the ranks is that we can provide such predicted values directly to a service broker or service client who may have their own selection and ranking methodology.

Though we worked with data clustering services, we expect that our approach to work for other data processing or analytic services. Our proposed system, once implemented in a service search engine or a service marketplace, allows the QoS data of various data analytic services for datasets to be collected in an implicit manner, as opposed to the approach taken in the experimental database methodology [42] which relies on researchers to submit their experiment information.

CHAPTER 3

METHODOLOGY

3.1 Introduction

Service selection mechanisms can be based on syntactical, semantic and structural information. Services that provide certain functionality can be found using service registries such as UDDI implementations through service selection systems. For example, a user may require a data storage service and identify several services through a keyword search. However, the candidate services may not fulfill the user's concerns about the quality of the service, such as the availability of the service, the security the service offers and the cost of using the service. Thus, in spite of having retrieved various data storage services using a functional matching technique (keyword search in this case), the non-functional aspects of services need to be taken into consideration as well to ensure better selection of services. QoS driven selection techniques can be applied on functionally matched services to filter services based on the QoS requirements of a client.

Several QoS based selection techniques have been discussed in Chapter 2. Some methods use vector based mechanisms to rank services [6-8]. Many recent works have incorporated user and system constraints with the help of optimization methods [25, 26]. Multi-criteria based approaches using AHP [12] or skyline concepts [13, 14] have also been proposed. However, such selection mechanisms may not suffice in selecting services that process data such as data analytic services. Data analytic services can include data mining services, data cleansing services, data conversion services, etc. Since data needs to be processed by the service, the service behavior can vary for different types of data – for example, some services cannot handle categorical data or the processing time of using the service can increase for larger datasets.

Considering that datasets are now getting bigger and the rate at which they are being generated is also increasing, clients can no longer rely on locally installed data analytic tools to process data and thus may need to consult web-services. Hence, data analytic algorithms can be implemented as services to provide their functionality across the web while ensuring interoperability among the various entities that need to communicate for this process.

In a scenario where such data analytic services are used by clients, a data dependent QoS based service recommendation can assist the user in identifying services that are most appropriate for the input dataset. In this thesis, we consider the problem of selecting data analytic services such as data mining services based on the input dataset's characteristics, i.e., the meta-attributes of the dataset. The service selection methodology would need to accommodate for different types of datasets.

3.2 Our Methodology

In our system, we consider data clustering services as an example of data analytic services. Clustering is a data mining method that discovers groups within a dataset in such a way that members within a group are similar to each other but are dissimilar to members belonging to other groups.

Our selection methodology can be applied in systems that offer data analytic web services such as a service marketplace. Our selection system can be initially trained with sample datasets or synthetic datasets to create a data repository with different meta-attributes. This repository can be improved over time as more datasets are used. The repository can be built with the collected dataset information (i.e., the meta-attributes) and corresponding QoS data of invoked services.

By exploiting this data repository, we can recommend services for a new dataset. This can be achieved by identifying datasets similar to the new dataset to predict the QoS values for the clustering services on a new dataset.

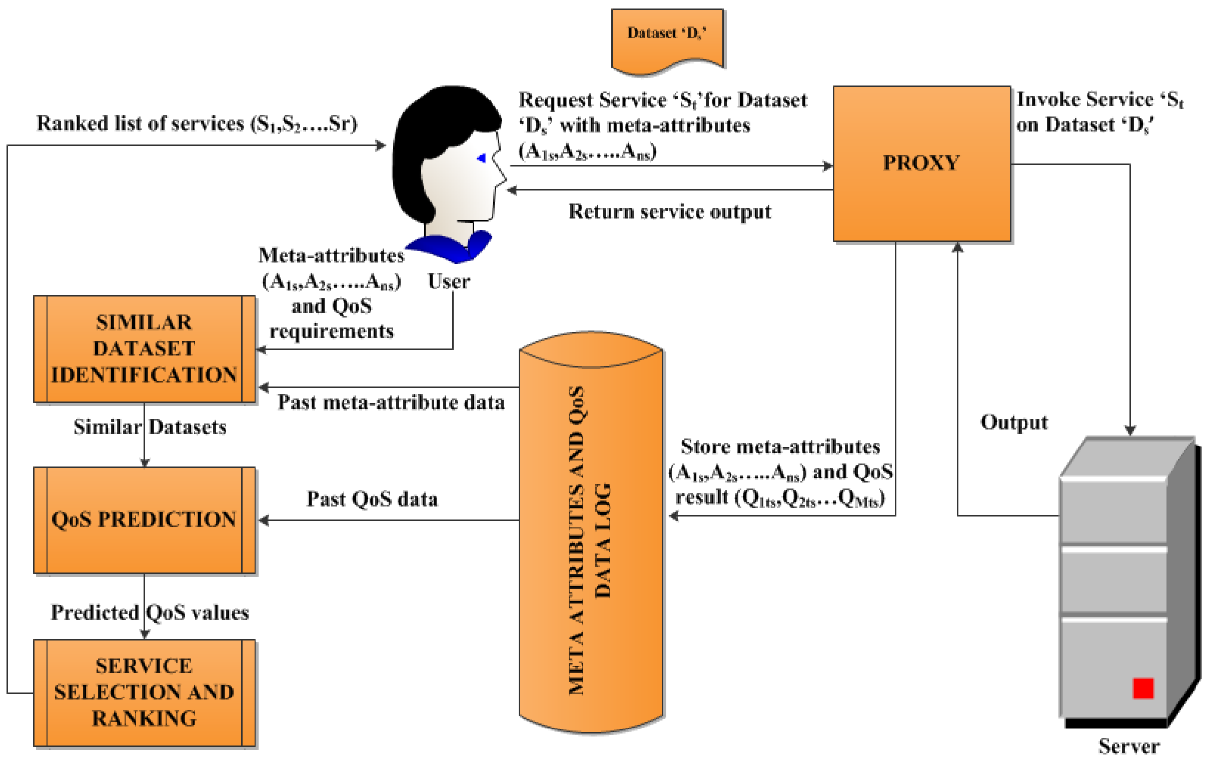
As shown in Figure 3-1, a user can seek a list of ranked services to make a decision about which service to apply on his dataset. To obtain the ranked list, the user first provides the meta-attributes (A_{1s}, A_{2s}, … A_{ns}) of his dataset 'D_s' where 'n' is the number of meta-attributes. The data dependent selection system consults the existing data repository to identify datasets similar to the user's dataset. The similar datasets can be discovered with the help of a similarity measure which computes similarity between datasets based on the meta-attributes describing the datasets.

As mentioned earlier, data-dependent QoS values may be "per dataset data-dependent QoS attributes" or "per service data-dependent QoS attributes". The "per dataset data-dependent QoS attributes" can vary according to the dataset involved. For instance, the latency of a service may change according to the size of the dataset. These QoS attribute values can be predicted for the candidate services with the help of the past QoS data of these services on the similar datasets. In this way, the behavior of the services can be predicted without having to actually invoke the service on the dataset. The "per service data-dependent QoS attributes" such as reliability, are computed as an overall value based on the service's performance on all the datasets it has processed.

Once the QoS values are estimated for all the candidate services for the current dataset, they can be used for ranking the services. This ranked list of services (S_1, S_2, … S_r) is provided to the client. The client can then choose which service to use for his dataset.

The user can select the service 'S_t' that he would like to apply on his dataset 'D_s' and have the system invoke the service on his behalf. Once the requested service has been invoked successfully, the results are returned to the user. For every service invocation made, the system records the meta-attributes of the input dataset to keep a log of the meta-attributes of the dataset used. It also saves the corresponding “per dataset data-dependent QoS attribute” values of the service that has been used for that dataset that include QoS attributes such as latency and response time.

Also, the “per service data-dependent QoS attribute” values are updated for the service. These values are computed based on all the datasets used for this service. Reliability and throughput can be considered as “per service data-dependent QoS attributes”. This type of data-dependent QoS attribute reflects the services’ ability to deal with a variety of datasets. In this way the data repository is updated for the service invocation with the current meta-attributes (A_{1s}, A_{2s}, …, A_{ns}) and the set of ‘M’ QoS result (Q_{1ts}, Q_{2ts}, … Q_{Mts}) to reflect the performance of service S_t on dataset D_s.



Dataset 'D$_s$'
Ranked list of services (S$_1$,S$_2$….Sr)
Request Service 'S$_t$'for Dataset 'D$_s$' with meta-attributes (A$_{1s}$,A$_{2s}$…..A$_{ns}$)
PROXY
Invoke Service 'S$_t$ on Dataset 'D$_s$'
Return service output
Meta-attributes (A$_{1s}$,A$_{2s}$…..A$_{ns}$) and QoS requirements
User
SIMILAR DATASET IDENTIFICATION
Past meta-attribute data
Similar Datasets
META ATTRIBUTES AND QoS DATA LOG
Output
Store meta-attributes (A$_{1s}$,A$_{2s}$…..A$_{ns}$) and QoS result (Q$_{1ts}$,Q$_{2ts}$…Q$_{Mts}$)
QoS PREDICTION
Past QoS data
Predicted QoS values
SERVICE SELECTION AND RANKING
Server

Figure 3-1: Data Dependent Service Selection System

3.2.1 Steps Involved in the Methodology

The main steps in our methodology are explained as follows:

1. Identification of Similar Datasets

In this step, datasets similar to the current dataset from the repository of datasets can be discovered with the help of the dataset's meta-attributes, i.e., (A_{1s}, A_{2s}, …, A_{ns}) such as the dataset size, dimensionality, data type, etc. This step will be discussed in detail in section 3.3.

2. Prediction of the QoS values of the services

The "per dataset data-dependent QoS attributes" can be predicted based on the QoS values of the services used on the similar datasets. The prediction algorithm used is explained in Section 3.4. The "per service data-dependent QoS attributes" can be computed based on the service performance on all of the past datasets used for the service.

3. Ranking the candidate services

The candidate services can be ranked according to the estimated QoS values using a traditional QoS service selection technique. The ranking methodology is covered in Section 3.5.

3.2.2 Quality of Service Properties

QoS values usually include metrics such as response time, latency, throughput, reliability, availability, security etc. Domain specific QoS values pertaining to the service application can be used as well. For example, in [8], the QoS service selection module presented is discussed with regard to the phone services provisioning domain. The authors used business related criteria such as penalty and compensation rate as the domain specific QoS measures. In our case, we can include accuracy as another QoS measure to determine the performance of the data clustering services since the clustering algorithms have different approaches to clustering the data and thus may have different levels of accuracy. In general, accuracy is an important measure for many data mining applications. However, it is often ignored in QoS-based service selection systems.

As discussed previously, some QoS properties may be data-dependent while others may not be. We categorized the data-dependent properties as "per dataset data-dependent QoS attributes" and "per service data-dependent QoS attributes". Table 3-1 shows some examples of attributes under each of these categories.

Table 3-1: Types of QoS Attributes with examples

Per Dataset Data Dependent QoS Attributes	**Per Service Data Dependent QoS Attributes**	**Data Independent QoS Attributes**
• Accuracy • Latency • Response Time	• Reliability • Throughput	• Availability • Integrity • Security

We have computed latency and accuracy as "per dataset data-dependent QoS attributes" and reliability as a "per service data-dependent QoS attribute". The definitions of these QoS attributes are as provided below.

1. **Latency:** Latency is defined as the "time taken for the server to process a given request" [24]. For clustering web services, we consider latency as the time it takes the service to build the clustering model for a given dataset.
2. **Accuracy:** This QoS value is used to determine the quality of the clustering results. We have used a modified version of the 'Inaccuracy' measure provided by the Weka toolkit [44]. The Weka software measures inaccuracy of clustering algorithms as the ratio of incorrectly clustered instances to the total number of instances as given below. It is expressed as a percentage as given in the equation 3.1.

$$Inaccuracy = \left(\frac{N_w}{N_I} \times 100\right) \qquad (3.1)$$

where N_w represents the number of instances that are incorrectly clustered and N_I represents the total number of instances in a dataset.

We have modified this measure to take into account instances that are not clustered as well i.e. instances that have been ignored by the clustering algorithm. This may happen since the clustering algorithm considers some instances as “noise” and thus ignores these instances. Accuracy of a clustering service on a dataset is given in equation 3.2:

$$Accuracy = 100 - \left(\frac{(N_w + N_u)}{N_I} \times 100\right) \qquad (3.2)$$

where N_u is the number of instances that have not been clustered.

We have used the above definition of accuracy to measure the clustering quality as we have knowledge about the actual clusters since we are using synthetic datasets in our experiment. In case the ground-truth data is not accessible, we can use intrinsic measures to evaluate the clustering quality. The Silhouette coefficient, for example, can be used as an intrinsic measure to determine the compactness of a cluster. It calculates how close an object in a cluster is to the other members of the same cluster as compared to objects in other clusters [16].

3. **Reliability:** Reliability is a quality metric that takes into account the proportion of service invocation failures to the total number of invocations. We have defined it as the proportion of the number of datasets for which the service did not fail to the total number of datasets used with the service. In this framework, we have accommodated failures due to internal algorithm issues and web services being unable to process datasets with certain characteristics such as a service not being able to handle numeric datasets or handling large dimensions. The formula is given below in equation 3.3, and this metric is also expressed as a percentage:

$$Reliability = \left(\frac{M_V}{M_D} \times 100\right) \qquad (3.3)$$

where M_V is the number of datasets the service successfully processes and M_D is the total number of datasets used by the service in the past.

3.3 Calculating similarity between datasets

We calculate the similarity between datasets based on their meta-attributes. Since clustering algorithms offer a different kind of performance for different types of dataset characteristics, we identified the following list of meta-attributes:

1. A_1: Dataset size – The size of the dataset or number of instances in a dataset.
2. A_2: Number of dimensions– The number of attributes in a dataset.
3. A_3: Percentage of Missing Data – The amount of missing data present in a dataset computed as a percentage.
4. A_4: Data type– This meta-attribute is used to indicate the type of data in the dataset. Currently we only consider two types – numeric and nominal, due to the constraints of the data generator we are using in the experiment.
5. A_5: Data distribution pattern – This meta-attribute is used to indicate what kind of distribution pattern defines the cluster positions. The clusters may be randomly placed or have a grid pattern. Though there could be more patterns, we consider these two in the current work.

Each dataset 'D_s' can thus be represented as a vector of its meta-attributes (A_{1s}, A_{2s}, A_{3s}, A_{4s}, A_{5s}). An example of a dataset's meta-attribute vector could be : (1000, 3, 5, "Numeric", "Random") which represents a dataset with 1000 instances, 3 dimensions, 5% missing data with a completely numeric dataset and randomly placed clusters.

The main steps for computing similarity values are presented in Algorithm 3.1. In this algorithm, we compute the similarity between the user input dataset and each dataset available in the repository. From our preliminary results, we found that the data distribution pattern and data type have the highest impact on prediction accuracy. Hence, we first compare if both datasets are of the same type i.e. if both are either numeric or nominal and also check if the pattern of the clusters is the same.

If the datasets match based on the aforementioned comparisons, we proceed to calculate the distance between each pair of the numeric meta-attributes that can be quantified. In our list of meta-attributes, dataset size, number of dimensions and percentage of missing data are the numeric meta-attributes. We use Manhattan distance similar to the one in [39] to calculate the distance between the meta-attributes as shown in equation (3.4):

$$Distance(Inp, Rep) = \sum_{\mathrm{i=1}}^{N_A} \frac{|\mathrm{V_{i.Inp}} - \mathrm{V_{i.Rep}}|}{|\mathrm{MAX_i} - \mathrm{MIN_i}|} \qquad (3.4)$$

Where $\mathrm{V_{i.Inp}}$ and $\mathrm{V_{i.Rep}}$ are the values for the $\mathrm{i^{th}}$ meta-attribute of the user input dataset and repository dataset respectively. $\mathrm{MAX_i}$ and $\mathrm{MIN_i}$ are the maximum and minimum values for the $\mathrm{i^{th}}$ meta-attribute among all the datasets. N_A is the number of numeric meta-attributes.

The algorithm for computing the similarity between the input dataset and repository dataset is provided below. As shown in, in line (12) the distance value is converted to similarity. Also, weighted similarity is computed as given in this line to incorporate weight values for the meta-attributes. The overall similarity between a pair of datasets is thus calculated as the weighted average of the similarities for the meta-attributes.

Algorithm 3.1: Pseudo code for calculating similarity between datasets

```
Input: Dataset DS_Inp- Input Dataset;// ‘Dataset’ is a user defined class
       Dataset DS_Rep- Dataset from existing repository;
        // ‘Meta-Attribute’ is a user defined class
       ArrayList<MetaAttribute> Meta_Attributes- List of numeric meta- attributes

Output: The overall similarity between DS_Inp and DS_Rep ;

Algorithm:
    (1)  Similarity(DS_Inp, DS_Rep, Meta_Attributes)
    (2)  {
    (3)          double overall_similarity = 0;
                 //now check if data type (‘type’) and data distribution pattern (‘pattern’) is same for both datasets
    (4)          If (DS_Inp.type < > DS_Rep.type || DS_Inp.pattern < > DS_Rep.pattern)
    (5)          {
    (6)                  return overall_similarity;
    (7)          }
    (8)          total_weight=0;
    (9)          For (MetaAttribute a : Meta_Attributes) // each meta-attribute 'a' in Meta_Attributes
    (10)         {
                         //calculate normalized Manhattan distance
    (11)                 current_distance = NormalizedDistance(DS_Inp.a, DS_Rep.a);
    (12)                 current_similarity = ( 1 - current_distance) * a.weight ;
    (13)                 overall_similarity += current_similarity;
    (14)                 total_weight += a.weight;
    (15)         }
    (16)         overall_similarity  = overall_similarity / total_weight ;
    (17)         return overall_similarity;
    (18) }
```

3.4 Prediction of QoS Values

After obtaining similarity values between the input dataset and datasets in the existing repository, we can now predict the “per dataset data-dependent QoS attribute” values of the dataset. To estimate these QoS values, for a particular service on the input dataset, we use a weighted sum approach often used in collaborative recommendation systems.

In recommendation systems, several techniques are used to predict the utility of an item for a user. As mentioned in [45], one technique of estimating the utility of an item for a particular user ‘c’ is to consider the similarity between the user *c* and past users as well as the rating given to that item by similar past users. One of the formulae listed in this paper is the weighted approach shown in equation (3.5) which sums the product of past user similarity, *Sim* and the corresponding user's rating for the item *s*. The predicted rating for user *c* on item *s* is given as:

$$R_{c,s} = n_f \sum_{c \in C} Sim\ (c, c')\ X\ R_{c',s} \qquad (3.5)$$

where $Sim\ (c, c')$ is the similarity between the user c and c' where c' belongs to the set of similar users 'C'. $R_{c',s}$ is the rating given to item *s* by the user c'.This aggregated value is then multiplied by a normalizing factor n_f, which is usually defined as the inverse of the sum of the similarity values used. This way, the calculation allows a higher preference for more similar users.

We have adopted this approach to predict the "per dataset data-dependent QoS" values that include latency and accuracy for each service on the new dataset. This method is shown in equation 3.6 with QoS value used in lieu of the rating values and similarity between datasets is computed instead of similarity between users.

$$QoS_{S_t,D_{Inp}} = n_f \sum_{D_{Rep} \in D_Z} Sim(D_{Rep}, D_{Inp}) \quad X \quad QoS_{S_t,D_{Rep}} \ (3.6)$$

In the above equation (3.6), we compute the QoS value, $QoS_{S_t,D_{Inp}}$for the candidate service S_t with the input dataset, DS_{Inp}. The similarity is computed for the set of similar datasets D_Z . The normalizing factor in our case would be given in (3.7):

$$n_f = \frac{1}{\sum_{D_{Rep} \in D_Z} Sim(D_{Rep}, D_{Inp})} \qquad (3.7)$$

To identify the similar datasets, we use a threshold, "Threshold_Similarity" (for example 0.9) to select datasets with similarity above this threshold value. Another approach to select similar datasets is to choose the top 'K' similar datasets.

We have also provided an automatic selection of the similarity threshold value ("Threshold_Similarity"). Our approach is to first identify the similarity value of the most similar dataset (with respect to the input dataset) which we refer to as the “highest similarity value”. For example for a new dataset, the closest neighbour dataset may have a similarity value as 0.84. We then set the threshold value to the highest value that is a multiple of 0.05 and is smaller than the “highest similarity value”. In this example, the threshold value would be set as 0.8.

We have also accommodated for failed services in this algorithm. Another parameter, ‘”Threshold_Fail” assists us in predicting if the service fails for the input dataset. For example, this threshold value can be set to 50% as such a value seems to be reasonable enough to predict if a service may fail, if it has failed for half or more than half of the time for similar datasets. For a new dataset, if there are 5 similar datasets identified, 3 of which could not be used successfully with service ‘S’ i.e. 60% of the datasets could not use the service, we predict that service ‘S’ to fail for the current dataset as well.

In case there are no similar datasets identified for the new dataset, we consider the average QoS values of the available service for all the past datasets. If a new service S_q is introduced in the system, we can build an initial repository of QoS values by invoking it on a set of synthetic datasets or sample datasets. We could even use some of the existing datasets in the repository. To decide which datasets to invoke, we can first randomly pick a dataset D_g and compute the QoS values for S_q with D_g. The next dataset D_h can be randomly picked and the service S_q is invoked for this dataset as long as D_h is not “similar” to D_g based on a pre-decided similarity threshold value. We try to use a variety of different datasets and avoid using too many similar datasets for this initial training phase. The pre-decided similarity threshold value can be set as the most frequently used similarity threshold value used in the system.

Algorithm 3.2: Pseudo code for prediction of QoS values

```
Input: Dataset DS_Inp-Input Dataset;
        double Threshold_Fail- The threshold for determining if a service may fail
        ArrayList<Service> Services-List of candidate services // ‘Service’ is a user defined class
        //’QoSMetric’ is a user defined class
        ArrayList<QoSMetric> QoSMetricList_perDS – List of per dataset data-dependent QoS metrics
        ArrayList<Dataset> SimilarDatasets- List of similar datasets identified

Output: The predicted QoS values for each service

Algorithm:
    (1)  Prediction (DS_Inp, Threshold_Fail, Services, QoSMetricList_perDS, SimilarDatasets)
    (2)  {
    (3)          double partialQoS, normalizingFactor;
    (4)          int count_failedServices;
    (5)          For (Metric M : QoSMetricList_perDS )  // each QoSMetric 'M' in QoSMetricList_perDS
    (6)          {
    (7)                  For (Service S : Services)// each Service 'S' in Services
    (8)                  {
    (9)                          partialQoS = 0;
    (10)                         normalizingFactor = 0;
    (11)                         count_failedServices = 0;
    (12)                         For(Dataset DS_Sim: SimilarDatasets)//each Dataset DS_Sim in SimilarDatasets
    (13)                         {
                                         //-1 represents failed services
    (14)                                 If(QoS value ‘V’ for service 'S' with DS_ Sim == -1)
    (15)                                 {
    (16)                                         count_failedServices++;
    (17)                                 }
    (18)                                 Else
    (19)                                 {
    (20)                                         partialQoS += ( DS_ Sim.Similarity(DS_Inp) x V )
    (21)                                         normalizingFactor += DS_ Sim.Similarity (DS_Inp);
    (22)                                 }
    (23)                         }
    (24)                         If(Percentage (count_failedServices) >= Threshold_Fail)
    (25)                         {
    (26)                                 QoSPredicted = -1;
    (27)                         }
    (28)                         Else
    (29)                         {
    (30)                                 QoSPredicted = partialQoS / normalizingFactor ;
    (31)                         }
         //This function records the predicted QoS value for metric ‘M’ of Service ‘S’ with ‘DS_Inp’
    (32)                 RecordQoSValues(DS_Inp, M, S, QoSPredicted);
    (33)                 }
    (34)         }
    (35) }
```

The algorithm for predicting QoS values is provided in algorithm 3.2. After the QoS predicted values are computed, the system is updated with each predicted QoS metric value for every service. We updated these values as a batch insert.

To identify the list of similar datasets, we used one of the two methods described previously, i.e., by selecting top K datasets or selecting datasets with similarity above the Threshold_Similarity.

3.5 Ranking of Web Services

After obtaining the predicted QoS values for the candidate services on the input dataset, our final task is to rank the services to help the user to select a service for the input dataset. The ranking method used here combines multiple QoS attributes.

The tendency of a QoS metric described the direction of the metric value. It may be positively or negatively monotonic. Metrics such as availability, reliability, security and accuracy have positive tendencies as a higher value for such metrics signifies a better performance of the service. However, metrics such as latency and response time have a negative tendency since services that run and respond faster are considered to perform better.

The users can additionally provide a priority value for each QoS attribute based on their preferences. For instance, a user may need a service that is time sensitive and thus give a higher priority to the response time. Another user may need services that are price sensitive and thus provide a higher preference value for the cost of using the service.

We have used a utility based approach to rank our services that combines the multiple QoS metrics involved. The utility function we have used allows different tendencies of QoS metrics and accommodates user preferences for different QoS values using weights. The utility function for Service S_r is given below in equation 3-8 and is based on the objective function in [26].

$$U(S_r) = \sum_{i \in X} \frac{\mathrm{V_{ir}} - \mathrm{MIN_i}}{\mathrm{MAX_i} - \mathrm{MIN_i}} * W_i \; + \; \sum_{j \in Y} \frac{\mathrm{MAX}_j - \mathrm{V_{jr}}}{\mathrm{MAX}_j - \mathrm{MIN_j}} \; * W_j \qquad (3.8)$$

where 'X' is the set of positively monotonic QoS metrics and 'Y' is the set of negatively monotonic QoS metrics. $\mathrm{V_{ir}}$ (or $\mathrm{V_{jr}}$) is the value of the QoS measure for service S_r. W_i (or W_j) represents the corresponding weight for the ith (or jth) QoS attribute. $\mathrm{MAX_i}$ and $\mathrm{MIN_i}$ (or $\mathrm{MAX_j}$ or $\mathrm{MIN_j}$) represent the maximum and minimum values for these QoS metrics among all the candidate services.

Based on the computed utility function solution, the services can be ranked. The algorithm for ranking the services is given in Algorithm 3.3. The solution value of the utility based expression is computed based on the QoS metric's tendency, weight, maximum and minimum values as per equation 3.8.

With the calculated solution values, services can be ranked. We have sorted the list of candidate services based on this solution value in a descending manner to have the service with the highest solution value on top of the ranked list. This list of ranked services is then presented to the user.

We have considered latency, accuracy and reliability as the QoS measures for service selection as discussed earlier. In our case, latency and accuracy values would differ for each input dataset provided by the user. The value for latency and accuracy (which are considered as "per dataset data-dependent QoS attributes") is predicted based on similar datasets as explained in section 3.4. For reliability, a “per service data-dependent QoS attribute”, the value of the service would change when a new dataset’s information is added to the repository.

Thus, we can predict ranking of candidate services for the user’s input dataset based on the dataset’s meta-attributes without having to compute the actual QoS values for all the services. As the system is used over time, it can learn to improve the service recommendation and ranking.

Algorithm 3.3: Pseudo code for ranking the services

Input: ArrayList<Service> Services-List of candidate services
ArrayList<QoSMetric> QoSMetricList– List of QoS metric attributes

Output: returns list of ranked services

Algorithm:

```
(1)  RankServices(Services, QoSMetricList)
(2)  {
(3)      For (Service S: Services) //each service S in ‘Services’
(4)      {
(5)          double SolValue = 0; //solution value
(6)          For (QoSMetric M: QoSMetricList ) // each QoSMetric M in QoSMetricList
(7)          {
(8)              double diff = getMaxValue(M) - getMinValue(M); //difference
(9)              If(M.tendency == Positive)
(10)             {
(11)                 SolValue+= ((getQoSValue(S,M) – getMinValue(M))/diff) * M.Weight;
(12)             }
(13)             Else
(14)             {
(15)                 SolValue += ((getMaxValue(M) - getQoSValue(S,M))/diff) * M.Weight;
(16)             }
             //update the Service object S’s member variable ‘SolutionValue’
(17)         UpdateSolutionValue(SolValue,S);
(18)     }
(19)     Services =SortServices_Descending(Services);// reorder services
(20)     return Services;
(21) }
```

3.6 Summary

In this chapter, we discussed our proposed methodology towards selection of services based on the dataset given by the user. This methodology can assist in selection of data analytic services. We have used data clustering services as example of such data analytic web services in our framework.

Our approach can be used to recommend a ranked list of services to the user based on his input dataset before he invokes an actual service. Our proposed approach can be used in a system that offers data analytic services such an existing service marketplace. For every service request with the input dataset, we record the meta-attributes (of the dataset) and the corresponding QoS information for the service. This recorded data can be maintained in our repository.

The system learns to recommend services based on such previously collected data. For this, we have identified 3 main steps. First, datasets similar to the current input dataset are identified from the existing repository. The meta-attributes of the input dataset can be used to determine which datasets from the repository are similar to it. Next, with the information of QoS values of previously invoked services on similar datasets, we can then predict "per dataset data-dependent QoS attribute" values for candidate services for the current input dataset. The "per service data-dependent QoS attributes" are computed based on all the datasets used for the service. Finally, we can rank the services based on these predicted QoS values using a service selection technique and present this list to the user.

CHAPTER 4

EXPERIMENTS

4.1 Introduction

In this chapter, we will first discuss our experimental methodology, followed by the implementation details and then proceed to our results. We would like to compare our prediction of QoS values as well as the service ranking with that of the traditional service selection methodology's results. We have worked with several datasets to compare these two cases. We have also analyzed the effect of the number of similar datasets and the impact of different meta-attributes used for prediction.

4.2 Experiment Design

We would like to prove that considering meta-attributes in the selection of services that process data is important. We have incorporated the use of some meta-attributes that can be used to determine the nature of the dataset for this purpose. In our experiments, we assume that the system has been used for a certain period of time, and that there has been a certain amount of QoS data collected from past datasets together with their meta-attributes. To verify our approach, we have generated a number of different datasets and used services that implement several data-clustering algorithms. This approach can be used to work with other data mining services as well, such as data classification and data regression services.

First, we generated various kinds of datasets with different meta-attributes, i.e. dataset size, number of dimensions, type of data (nominal and numeric), data distribution pattern and percentage of missing data. We used the 10-fold cross validation technique to split the datasets into training and test datasets. The training datasets have been used as the dataset repository for the system and the test datasets have been used to evaluate the system. We assessed the quality of the prediction of QoS values for the services on these test datasets. We also compared the ranking of the services based on predicted QoS values with respect to the ranking of services based on actual QoS values for the test datasets.

We compared our results with the default ranking which would be the average QoS-based ranking algorithm. Since previous service selection techniques do not consider the type of data being processed by the service, the QoS values in such a case would be the average of QoS values in the past invocations. To compute the average value for the QoS metric 'M' for a particular service 'S' we compute it as the average of all the values for 'M' over all the datasets used by 'S'.

We have used two approaches for identifying similar datasets to the input dataset as mentioned previously. One method is to select the top K datasets closest to the input datasets. We tested different values of K to examine how the prediction accuracy can vary for different values of K. The other approach is to select datasets with similarity above a specified threshold.

We have tested the effect of having different number of datasets with a common range of meta-attributes. Also, we have analyzed the impact of different meta-attributes on the prediction accuracy.

4.2.1 Dataset Generation

To generate datasets for testing the data clustering web services, we opted to use synthetic datasets as opposed to working with real datasets so that we can generate a wider variety of datasets based on their meta-attributes to train and test our system. This way we can systematically test the impact of different meta-attributes as well as their values on the accuracy of our selection system. Although the UCI repository provides real datasets for machine learning research, there are only a limited number of them for testing clustering algorithms. Most of the meta-learning papers use a handful of UCI datasets. In our opinion, using a small number of datasets is hard to study the impact of meta-attributes on the system performance. Therefore, in this work, we used an artificial data generator provided by the Weka [44] toolkit. This data generator is based on the generator developed by the authors who proposed the BIRCH Clustering algorithm [46]. This data generator places the cluster centres according to a pattern parameter provided by this data generator. We have used the grid pattern and random pattern to generate our clusters to distinguish between the results when different patterns are used. The grid pattern generates clusters with their centres placed on a $\sqrt{l} \times \sqrt{l}$ grid where l is the number of clusters. The cluster centres are placed randomly for the random pattern. The data points are generated based on a normal distribution. The original data generator supports the creation of only two-dimensional datasets. The Weka implementation further allows generation of datasets with higher dimensions.

The datasets generated using this generator have numeric values. To create nominal datasets, we applied the Weka's 'Discretize Filter' on the dataset. This filter is an attribute filter which can discretize a range of numeric dimensions into nominal attributes. This discretization is based on simple binning [44].

To generate datasets with missing values, we created a program to randomly delete values in the dataset. A percentage parameter is used to determine the number of values to be deleted. For each value to be deleted, a random number ranging from (1, N) is generated, where N is the total number of values (i.e. the product of number of instances and number of attributes). The value at this corresponding number is then removed.

In our experiments we used 5 meta-attributes i.e. “Dataset size”, “Number of dimensions”, “Percentage of missing data”, “Data Type”, “Data distribution Pattern”, first three of which are numeric attributes. For dataset size and number of dimension, we decided to consider different scale values such as “small”, “medium” and “large” while for missing data we considered two scales: “small” and “large”. These scales are based on different ranges that were determined according to our preliminary analysis. After having established different ranges as scales, we can easily work with different combinations of meta-attributes for the experiments. For missing data, we observed that for higher amount of missing data, the accuracy of some services decreased significantly. In many cases we observed that some services outperformed others for datasets with missing data above 10-15%. We used only 2 scales of missing data since datasets with a very high amount of missing data is generally not used. The data type can be “numeric” or “nominal”. We have used datasets of two distribution patterns – “grid” and “random” as explained above. The meta-attributes and the possible values are shown in Table 4-1.

Table 4-1: Meta-attributes

Meta Attribute	**Value**
Dataset size	Small, Medium or Large
Number of dimensions	Small, Medium or Large
Percentage of missing data	Small or Large
Data type	Nominal or Numeric
Data distribution pattern	Random or Grid

The range values used for datasets size, number of dimension and missing data scales are provided in Tables 4-2 to 4-4. We limit the size of the datasets to a maximum of 5000 as very large sizes slow down the experiment process. The datasets were generated with different number of clusters ranging from 4 to 6 clusters per dataset. The radius of the clusters ranged from 0.1 to 0.5 units.

Table 4-2: Range for ‘dataset size’

Dataset size scale	Number of instances
Small	500 - 1000
Medium	2500 - 3000
Large	4500 - 5000

Table 4-3: Range for ‘number of dimensions’

Number of dimensions scale	Number of dimensions
Small	5 - 20
Medium	30 - 45
Large	55 - 70

Table 4-4: Range for ‘percentage of missing data’

Missing data scale	Percentage of missing data
Small	0 - 5%
Large	15 - 20%

4.2.2 QoS Computation

In this set of experiments, we have used services created in [47] that wrap various clustering algorithms available through the Weka Toolkit API. These services have been implemented as RESTful services. By applying these services on the datasets, we can collect QoS values of the services for each dataset. The clustering algorithms used in these services for our experiments are listed in Table 4-5.

As mentioned previously, we have computed two “per dataset data-dependent QoS metrics” – latency and accuracy for all services for each dataset. Latency is computed as the time it takes the service to process the dataset. In this case, it is the time to build the clustering model for the dataset. Accuracy is calculated as the percentage of correctly clustered instances. We also computed one “per service data-dependent QoS metric” i.e. reliability. Reliability is determined as the percentage of number of datasets for which the service did not fail to the total number of datasets used with the service.

Table 4-5: List of Services

No.	Clustering Algorithm used a service
1	CLOPE
2	Cobweb
3	DBSCAN
4	EM
5	FarthestFirst
6	SimpleKMeans
7	XMeans
8	Hierarchical Clustering with SINGLE link
9	Hierarchical Clustering with COMPLETE link
10	Hierarchical Clustering with AVERAGE link
11	Hierarchical Clustering with WARD link

4.2.3 Experiment Settings

For most of our experiments, unless explicitly stated, we used the automated similarity threshold method (explained in section 3.4) to determine the similarity threshold to be used for prediction of the QoS values. This method selects the similarity threshold based on the “highest similarity value”, i.e. the similarity value of the most similar dataset to the input dataset. The threshold value is set to the highest value that is a multiple of 0.05 and is smaller than the “highest similarity value”. The other approach to prediction is to use the top K similar datasets which we will consider in one of our experiments.

We set the “Fail_Threshold” to 50% as we felt it is a reasonable value to consider for determining if a service may fail for the new dataset if half or more than half of the similar datasets could not be handled by the service. We used an equal weight distribution for all the numeric meta-attributes in the similarity computation. We also used equal weights for the QoS attributes in the service selection module.

We used the clustering services with most of the default settings. The minimum number of points in DBSCAN was set to the minimum number of points in a cluster. For k-means, X-means, Farthest First, EM clustering and the hierarchical clustering services, the number of clusters parameter was set to the actual number of clusters in the dataset.

As mentioned previously, we used the 10-fold cross validation method to assess our results. In this evaluation procedure, the datasets are randomly split into 10 samples. We use 10 rounds of validation with one sample as the test datasets for evaluation and other 9 as training datasets.

4.2.4 Evaluation Metrics

For the purpose of evaluating our experimental results, we used two metrics: MAE (Mean Absolute Error) and SRC (Spearman's Rank Correlation). We used MAE [48] to assess the accuracy of the prediction of the "per dataset data-dependent QoS metrics" (i.e. latency and accuracy). MAE measures how close the predicted results are to the actual values. We first normalized these QoS values i.e. latency and accuracy and then computed MAE. The MAE value for QoS attribute 'Q' is computed as given in equation 4.1.

$$MAE(Q) = \frac{1}{N_D \times N_S} \sum_{i=1}^{N_D} \sum_{j=1}^{N_S} |V_{A_{S_i,D_j}} - V_{P_{S_i,D_j}}| \quad (4.1)$$

where $V_{A_{S_i,D_j}}$ and $V_{P_{S_i,D_j}}$ represents the actual and predicted values for the QoS attribute 'Q' for service 'S_i' on dataset 'D_j'. The number of datasets is represented by 'N_D' , and 'N_S' is the number of services. Lower values of MAE i.e. values closer to 0 indicate better prediction accuracy.

The services are ranked based on their QoS values using a utility based approach as explained in section 3.5. This ranking method sorts services with the help of an objective function that combines all the QoS attributes. We use this ranking technique to rank the services based on their actual QoS values as well as based on the predicted QoS values. Since the traditional service ranking methodologies do not take meta-attributes of the dataset into account, the ranking would then be based on the average QoS values. We have generated a ranking based on the average QoS values for comparison purpose.

The SRC coefficient [41] can be used to assess the ranking accuracy by calculating the relationship between the rankings based on actual and predicted QoS values. We also computed SRC for the average QoS-based ranking which can be considered as the default ranking. SRC is computed for the ranked list of services for a dataset ‘D’ as shown in 4.2.

$$SRC(D) = 1 - \frac{6\sum_{i=1}^{p} d_i^2}{p(p^2-1)} \qquad (4.2)$$

where d_i is the difference between the ideal and predicted ranking value for the ith service. p is the size of the ranking which in this case would represent the number of services being ranked. The SRC coefficient can range from -1 to +1. A value close to 1 reveals that the two rankings are very similar.

4.2.5 Implementation Details

To implement our data-dependent service selection framework, the dataset generation process as well as the procedure for the evaluation, we used the JAVA language. The services were hosted on a local APACHE server. The configuration used to run these programs are as below:

CPU: Intel CORE i5

RAM: 8 GB

Operating System: Windows 7

Software: Eclipse Kepler Service Release 1 with Java and Matlab for preliminary analysis

Database: MySQL

APIs: Weka API for dataset generation and clustering and IBM CPLEX Optimization API for service selection.

4.3 Results Compared to the Average QoS based Ranking

With the different possible values of meta-attributes as shown in Table 4-1, we can have 3 x 3 x 2 x 2 x 2 = 72 combinations. For each of these combinations, we can generate a number of datasets. Table 4-6 shows the distribution of datasets in each combination for each of these pairs.

Table 4-6: Distribution of datasets

Number of datasets per combination	**Number of combinations**	**Total datasets**
30	4	120
20	8	160
10	12	120
5	28	140
1	20	20

For this experiment, we applied all the services on this set of 560 datasets. We compute the overall SRC by computing the average SRC for all the datasets. The overall SRC result is provided in Table 4-7. We compute the overall SRC as the average SRC over all the datasets. We measured the overall MAE by computing the average MAE over all the services on each dataset for accuracy and latency to assess our prediction quality as shown in Table 4-8 and 4-9. We compare our predicted result to the average QoS scenario.

Table 4-7: Overall SRC result

Case	Average SRC	Standard Deviation
Predicted Result	0.960	0.047
Average Result	0.769	0.109

Table 4-8: Overall MAE result on Accuracy

Case	Average MAE	Standard Deviation
Predicted Result	0.058	0.039
Average Result	0.403	0.069

Table 4-9: Overall MAE result on Latency

Case	Average MAE	Standard Deviation
Predicted Result	0.192	0.093
Average Result	0.784	0.066

We can see that our predicted SRC result is much higher than the average case. Our SRC result indicates that our prediction is quite close to the actual ranking result. Our MAE results for both latency and accuracy are also much lower than the average case. Latency has a higher variation among the datasets as latency increases with increase in the size of the dataset and number of dimensions. Thus the MAE value for predicting latency is higher than that of accuracy.

Once we provide the ranked list of services to the user, a user is likely to pick the highest ranking service for his dataset unless perhaps the service is unavailable for some reason. We compared the accuracy of predicting the highest ranking service as shown in Table 4-10.

Table 4-10: Rank 1 Prediction Accuracy

Case	Accuracy
Predicted Result	74.11 %
Average Result	35.36 %

As shown in the above table, our prediction accuracy is quite decent and much higher, more than twice as that of the average case. Thus, it can be seen that incorporating a meta-learner for prediction of QoS values and ranking is important. Relying on the traditional approaches with the average QoS values shall certainly not be sufficient and accurate for ranking data analytic services.

4.4 Effect of number of datasets

In this set of experiments, we created different dataset repositories based on the number of datasets in a combination. In total, we worked with 5 different repositories each with equal number of datasets per combination. We set the number of datasets in a combination as 1, 5, 10, 20 or 30 for each repository. For example, we created a dataset collection based on the combinations with only 10 datasets per combination.

In this manner, we created 5 different sets or collections of datasets. In each of these sets, the number of datasets per combination are equal, i.e. the first set or collection with 1 dataset per combination, the second with 5 datasets per combination and so on.

We worked with the same 20 combinations for all the repositories. The distribution of dataset combinations with respect to the numeric meta-attributes (i.e. dataset size, number of dimensions and percentage of missing data) is shown in Table 4-11. The 5 listed combinations were used for both types of data (numeric and nominal) and both patterns (random and grid), which collectively forms 20 different combinations.

Table 4-11: Meta-attributes used

Dataset Size	**Number of Dimensions**	**Percentage of Missing Data**
Small	Small	Small
Small	Medium	Large
Medium	Medium	Small
Medium	Large	Large
Large	Small	Large

Figure 4-1 reveals that the ranking accuracy represented by SRC values is the lowest for the repository with 1 dataset per combination while MAE values for both latency and accuracy were highest for this case as well. On the other hand, the rest of the repositories had similar results in terms of SRC and MAE values. From this we observe that when there is only 1 dataset in a combination, the accuracy of our prediction is affected. The other 4 cases have higher prediction accuracy since there are more similar datasets available.

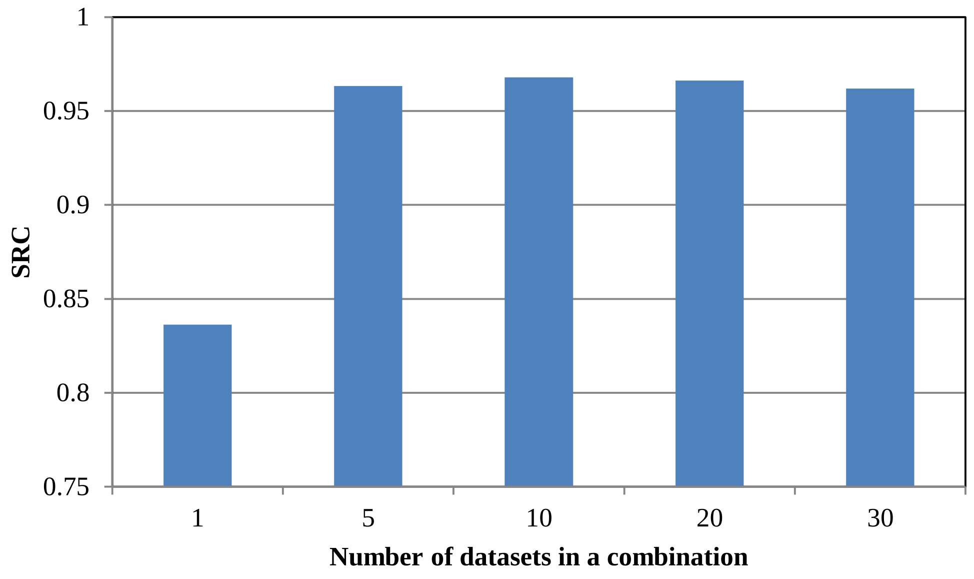


Figure 4-1: SRC values for different number of datasets in a combination in the repository

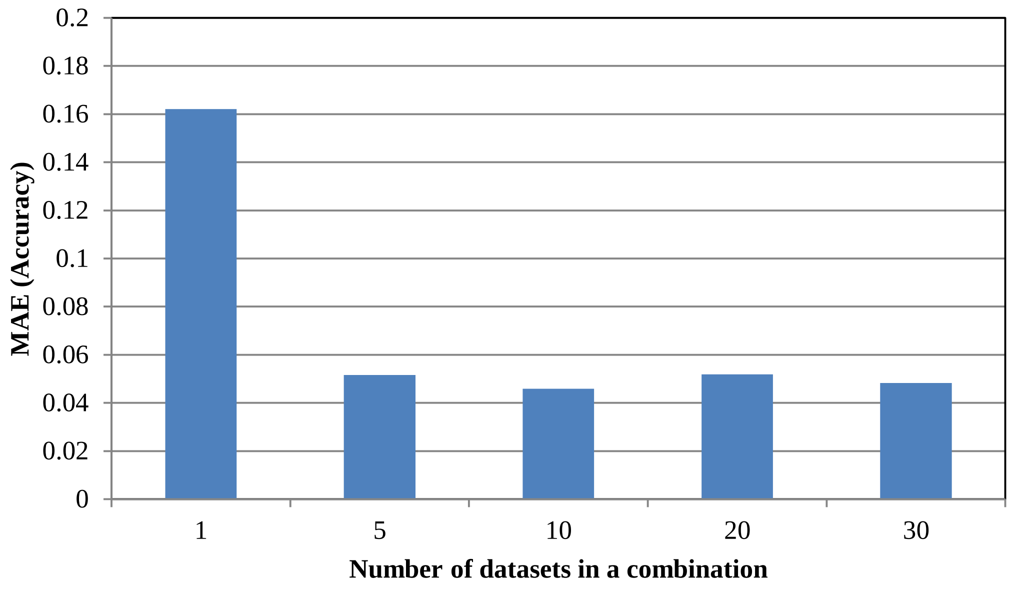


Figure 4-2: MAE (Accuracy) results for different number of datasets in a combination in the repository

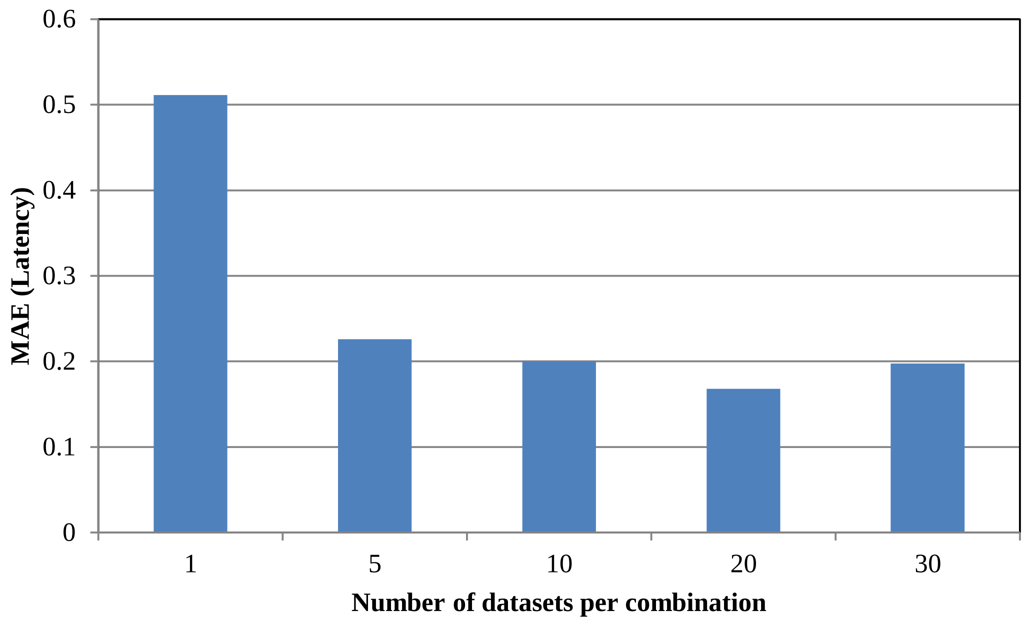


Figure 4-3: MAE (Latency) results for different number of datasets in a combination in the repository

4.5 Results for Top-K Similar Datasets

As discussed in section 3.4, to predict the “per dataset data-dependent QoS attribute” values, we first need to identify a set of datasets similar to the input dataset. We described two approaches to select similar datasets. One method is to use the top K most similar datasets and the other is to specify a similarity threshold to choose datasets with similarity above this threshold. The selection of the similarity threshold can be done automatically as well.

In this set of experiments, we analyzed the effect of using the top K similar dataset approach for prediction of QoS values. In this approach, the number of neighboring or similar datasets is selected by the user. We worked with different values of K (1, 5, 10, 15, 20, 25, 30, 35 and 40) on the set of datasets.

First, we used the same set of 560 datasets from the experiment described in section 4.3 to perform this experiment. It can be seen that the SRC is highest at the value K = 5 in Figure 4-4. For K greater than 5, the SRC value begins to decrease. We have a similar observation for the MAE results for latency and accuracy in Figures 4-5 and 4-6, i.e. the overall error is lowest at K = 5 and then the MAE increases after this point.

K = 5 seems to provide the best prediction results. Also, the prediction accuracy corresponding at K = 1 and K= 10 are not very different from the results at K = 5. We compared these results with the automated similarity threshold selection technique, i.e. the original results mentioned in section 4.3, and found that the difference in the original MAE and SRC results with respect to K = (1, 5, 10) is not significant.

Also, a closer look at the automated similarity threshold selection technique showed that smaller values of K were selected by this module for many datasets. For about 40% of the datasets, K was chosen as either 1 or 2 by this technique as shown in Figure 4-7. Though K = 5 may seem to provide the most optimal result as previously observed, we noticed that in many cases, different values of K chosen by the automated similarity threshold selection module provided a better SRC value. We investigated the combinations which have 5 or more datasets per combination and analyzed these datasets. We compared these results with the results at K=5. We found that the automated selection method provides the same result as that of K=5 for 55% of these datasets. For 18% of the remaining datasets, the automated selection technique had higher accuracy than that of K=5. Hence, for a user unsure of which K value to use, the automated similarity threshold selection technique should be adequate to provide a reasonable, if not optimal result.

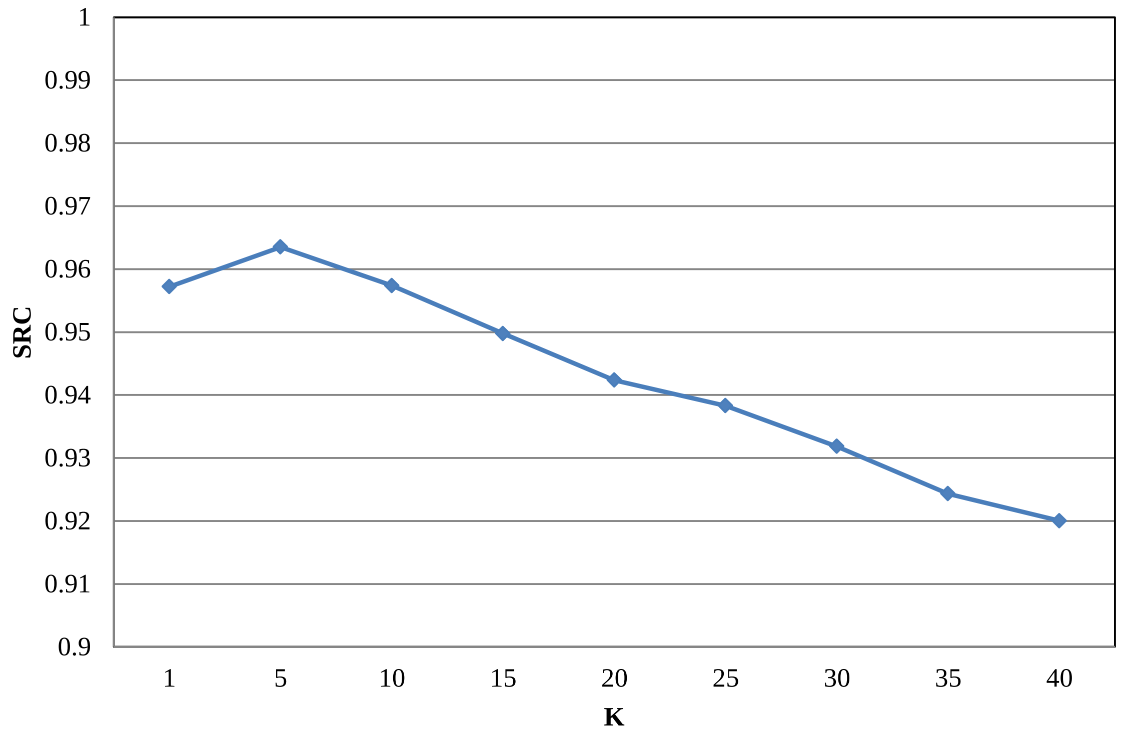


Figure 4-4: Effect of K on SRC

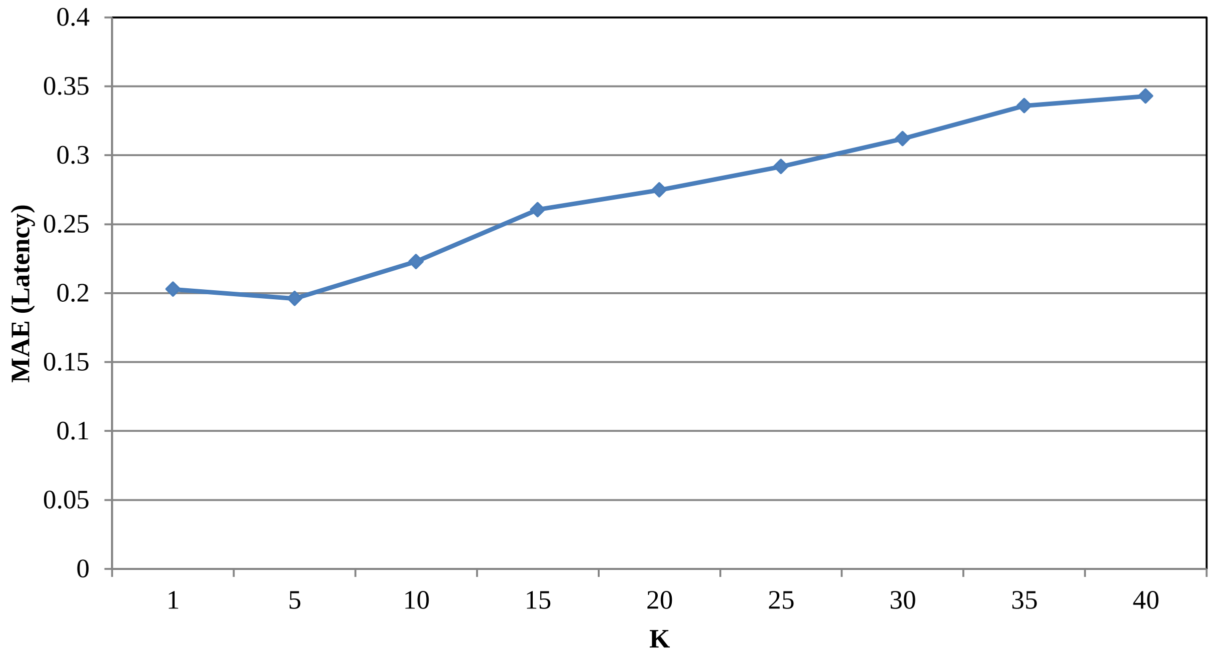


Figure 4-5: Effect of K on MAE (Latency)

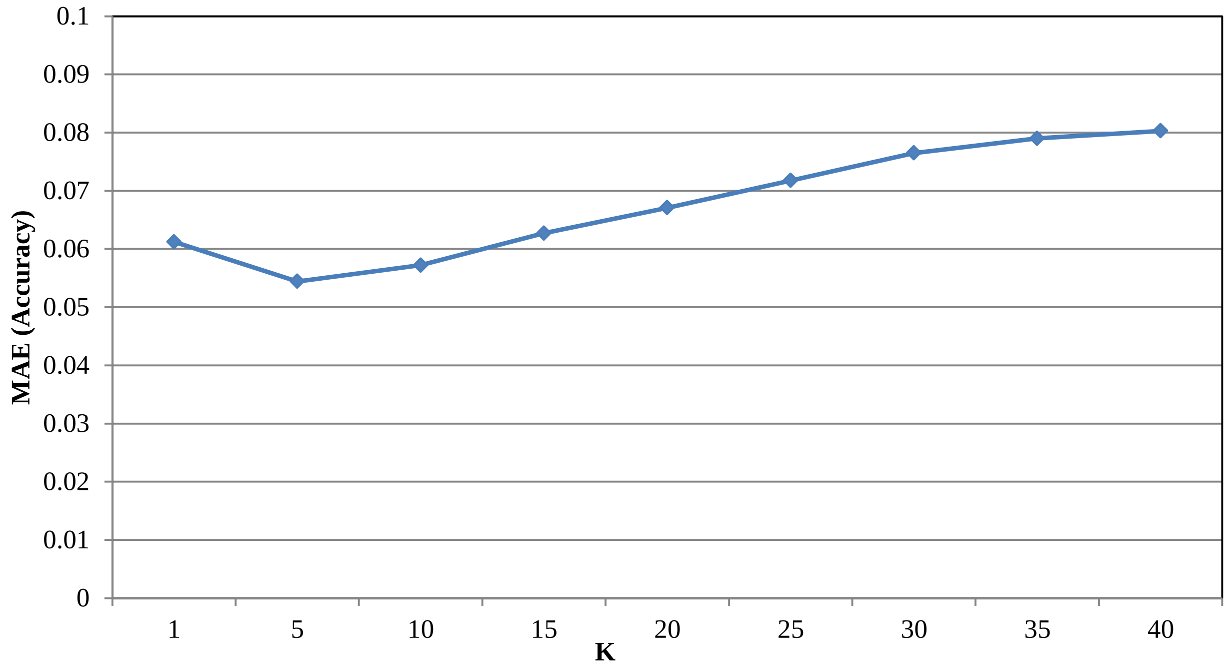


Figure 4-6 : Effect of K on MAE (Accuracy)

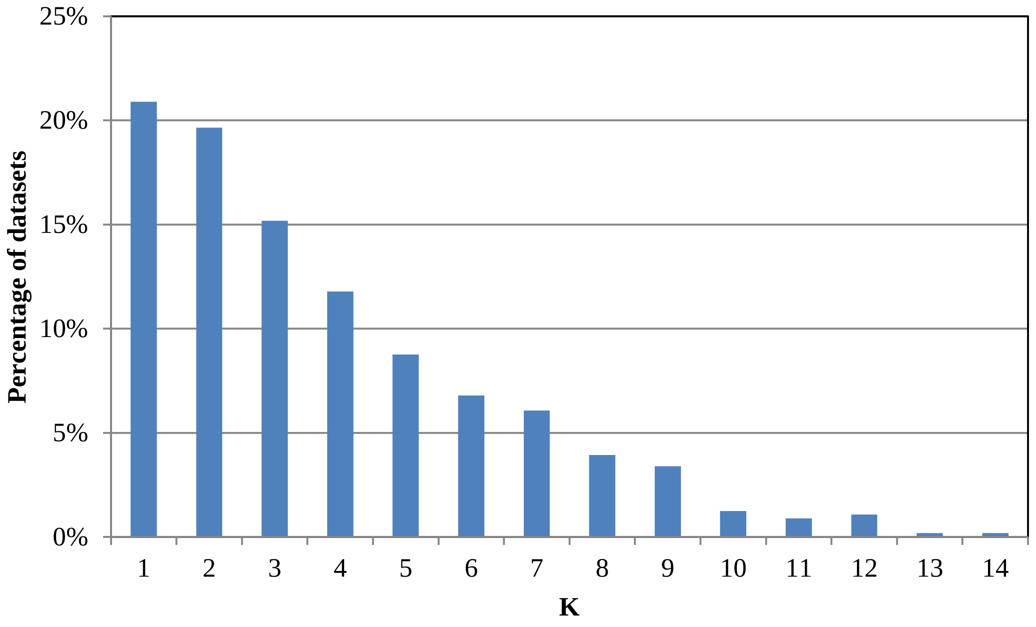


Figure 4-7: Value of K picked by the automated similarity threshold module

We also worked on the data repositories with 5, 10, 20 and 30 datasets per combination from section 4.4 to study the impact of increasing the value of K on our prediction results for each of these repositories. We denote the number of datasets in a combination as N_C. The logic behind considering these separate groups is we expect that the datasets that are identified to be similar to the test dataset should belong to the same combination as they are based on the same range of meta-attribute values. The SRC and MAE results for these 4 cases are as shown in Figure 4-8 to 4-10.

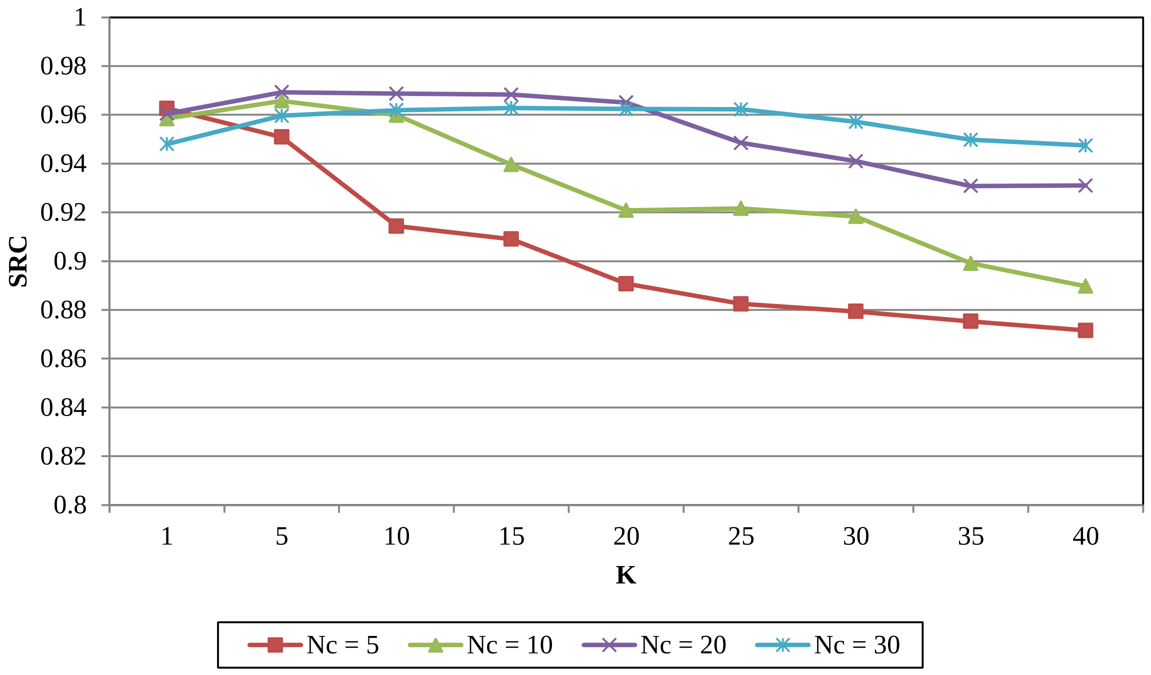


Figure 4-8 : Effect of K on SRC based on N_c

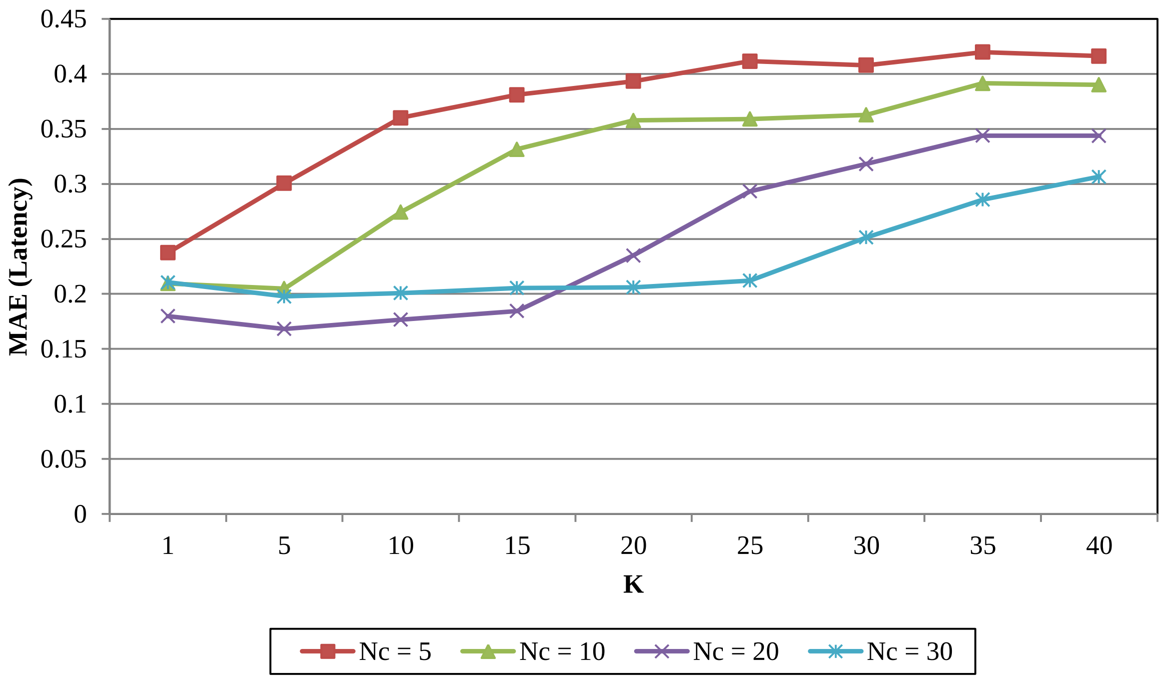


Figure 4-9: Effect of K on MAE (Latency) based on N_c

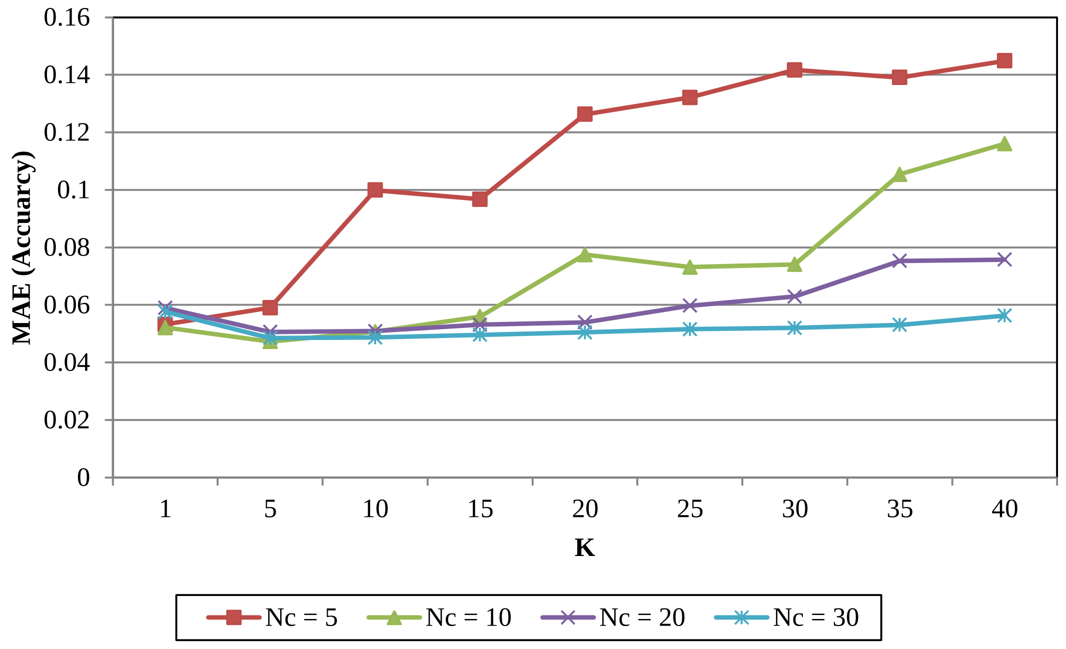


Figure 4-10: Effect of K on MAE (Accuracy) based on N_c

In the graphs shown above, we can see that lower values of K usually yield better prediction results. For values of K less than N_C, we can see that the increase in SRC or decrease in MAE is not significant. The top K (with K less than N_C) datasets for the input dataset have been found to usually belong to the same combination and thus the prediction results in terms of SRC and MAE are quite consistent till this point. However, for K values above N_c, the trend for SRC is a decreasing one while the MAE values increase after this point.

The effect of increasing the value of K on the prediction accuracy for N_c = 20 and 30 is less than for the cases where N_c is 5 or 10 since it is more probable to have more similar datasets in the same combination as these datasets.

If we compare the "best" SRC prediction results for each of these 4 repositories and with the prediction results provided by the automated similarity threshold technique, we find that the SRC results are quite close. The automated similarity threshold technique tends to pick smaller K values. Figure 4-11 shows the values of K chosen by this automated mechanism. Thus we can conclude from this set of experiments that lower values of K generally provide better prediction results.

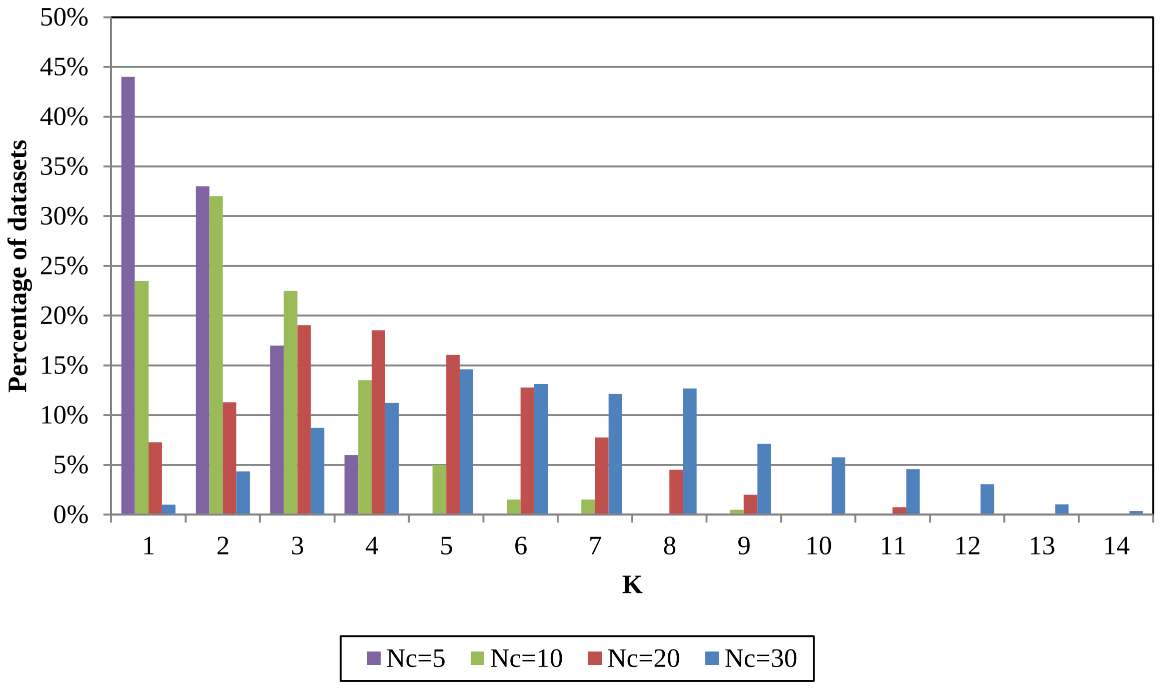


Figure 4-11: Value of K picked by the automated similarity threshold module

4.6 Impact of meta-attributes

Similarity between datasets is based on the set of meta-attributes as explained in section 3.3. In this series of experiments, we examined the influence of each of the meta-attributes by eliminating one meta-attribute at a time in the similarity computation for each of these experiments. We worked with the same set of 560 datasets mentioned in section 4.3. We compare 6 cases, the first with all the meta-attributes considered and the next 5 with removal of a single meta-attribute in each case. The cases are listed on Table 4-12.

Table 4-12: Cases used to test impact of meta-attributes

Case	Meta Attributes Used
All	All Meta-attributes (original case)
Without Dim	All except number of dimensions
Without DS	All except dataset size
Without MD	All except percentage of missing data
Without DT	All except data Type
Without DDP	All except data distribution pattern

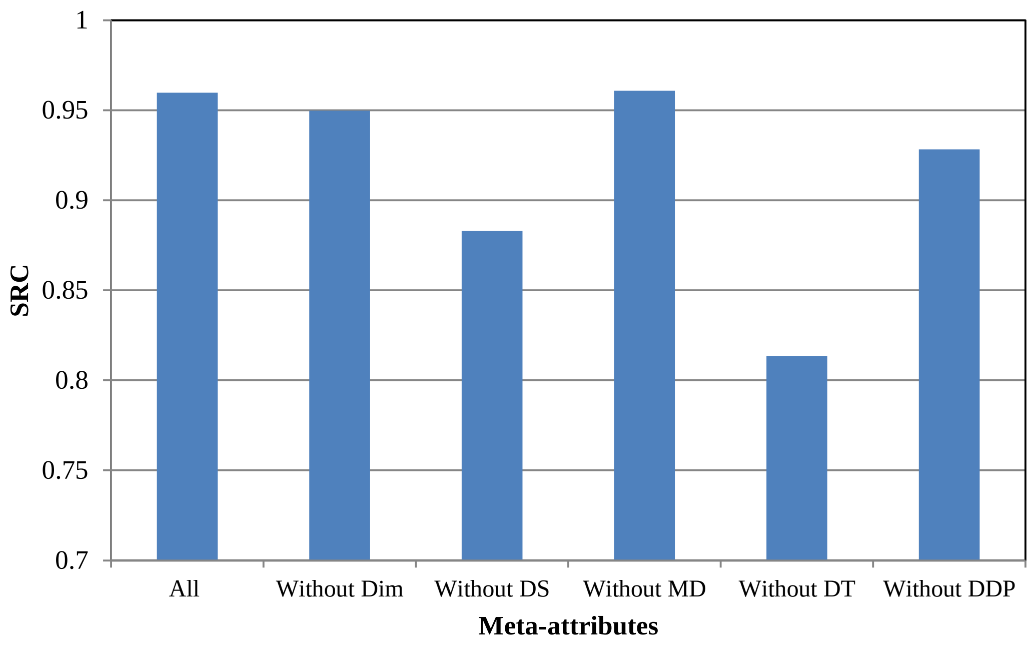


Figure 4-12: SRC after removal of meta-attributes

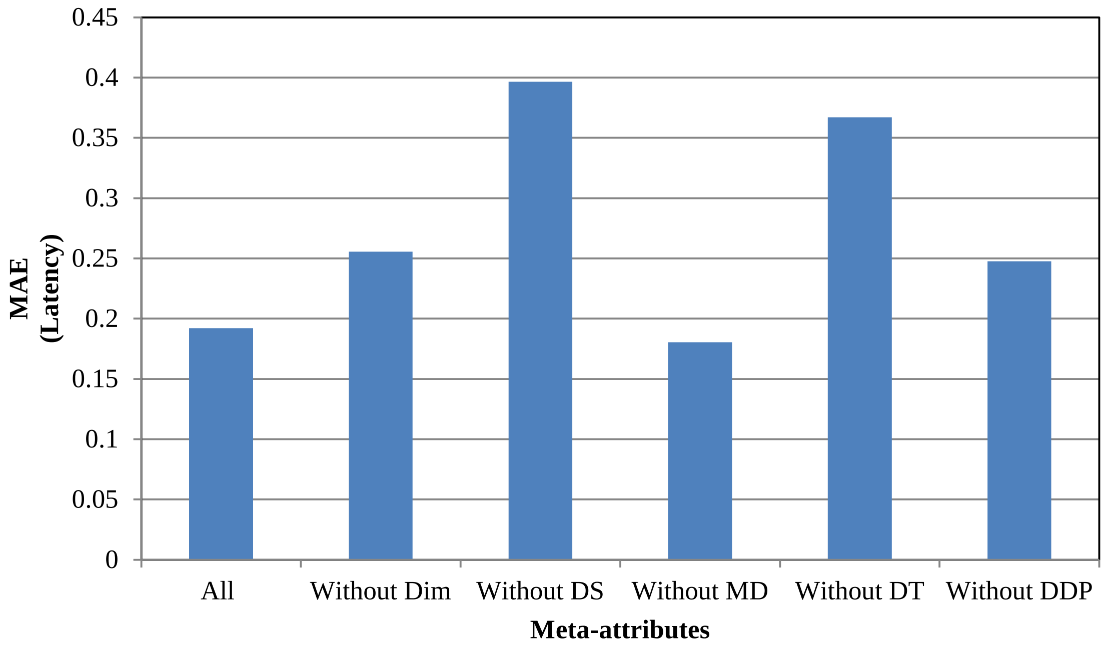


Figure 4-13: MAE (Latency) after removal of meta-attributes

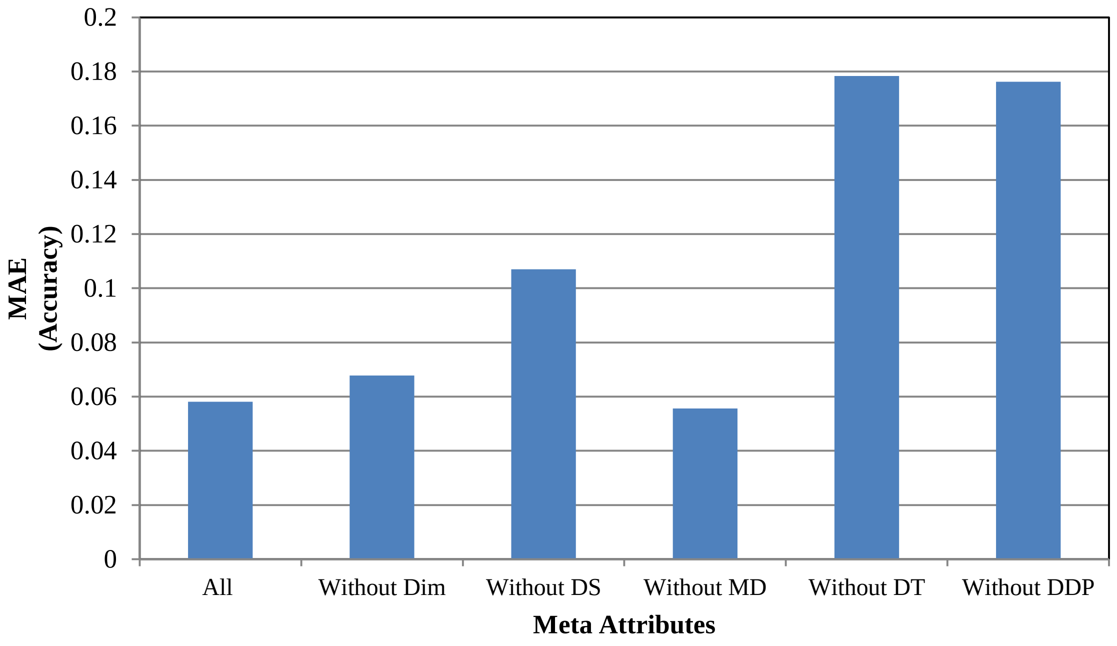


Figure 4-14: MAE (Accuracy) after removal of meta-attributes

The figures 4-12 to 4-14 show the effect of removal of each meta-attribute on SRC and MAE (for accuracy and latency) respectively. Eliminating either data type or data distribution pattern as a meta-attribute has a significant impact on the prediction of accuracy values (Figure 4-13). Services have different behaviors for different data distribution patterns. Through further analysis, we observed that most clustering services such as Farthest First and hierarchical clustering services did not perform very well on the grid pattern in terms of accuracy. On the other hand, the K-Means and EM clustering services are not affected significantly by numeric grid datasets as compared to random datasets.

Some services, for example Cobweb and X-Means clustering services cannot handle nominal datasets while Clope cannot handle numeric datasets. Using data type as a meta-attribute can assist in predicting these behaviors. For services such as the hierarchical clustering with single and average link, it takes much longer to process numeric data. This makes identifying the data type important for the prediction process.

The size of the dataset can influence the time it takes to process the dataset especially for services that use Hierarchical clustering to a very large extent. The large nominal datasets, for instance, can take about 50 times longer compared to the smaller datasets to process for hierarchical clustering services with single, complete and average linkages. Such algorithms do not scale well considering that their clustering algorithm have a high complexity.

Certain services cannot handle high dimensions, for example Hierarchical clustering with Ward Linkage cannot handle datasets with a large number of attributes. The latency time increases two to four times for services such as K-Means, EM, Farthest First, X Means and DBSCAN.

It may seem that considering percentage of missing data as a meta-attribute has negatively affected the prediction results, though it can be noted that this negative influence is extremely low. We can however still use percentage of missing data as a meta-attribute as it impacts some services significantly. We have observed that the accuracy can decrease of a clustering service can decrease by about 30 to 40% for hierarchical clustering with single and ward linkage clustering services for datasets with "large" missing data (i.e. 15- 20% missing data). We also noticed that the latency increases twice as much for "large" missing data for services such as EM clustering, Clope and all the hierarchical clustering services for many nominal datasets.

As shown in Table 4-13, we have considered an example of the effect of data type on prediction by comparing the actual ranked list of services for a pair of datasets D_1 and D_2, the former with numeric data and the latter with nominal data respectively. Both datasets have the same combination for the rest of the meta-attributes, i.e., the values for (Dataset size, Number of dimensions, Data distribution pattern, Missing data) are (Small, Small, Random, Small). Clope cannot handle D_1 and DBSCAN, Xmeans and Cobweb cannot handle D_2.

Table 4-13: A ranking case example

Service Rank	Top services for D1	Top services for D2
1	KMeans	EM
2	FarthestFirst	KMeans
3	EM	FarthestFirst
4	Xmeans	Hierarchical Clustering with Average Linkage
5	Hierarchical Clustering with Complete Linkage	Hierarchical Clustering with Complete Linkage
6	Cobweb	Hierarchical Clustering with Single Linkage
7	Hierarchical Clustering with Average Linkage	Hierarchical Clustering with Ward Linkage
8	DBSCAN	CLOPE
9	Hierarchical Clustering with Ward Linkage	-
10	Hierarchical Clustering with Single Linkage	-

Thus, for such cases by eliminating certain meta-attributes, the ranking order of the services can change. Such trends can explain why the SRC value is also affected if a meta-attribute is not considered.

4.7 Summary

In this chapter, we explained our experiment settings and the process used for dataset generation. We described our implementation methodology and how we computed the QoS values for the services. We have also discussed the evaluation metrics, i.e. SRC and MAE that have been used to asses our results. We have compared our result to the average QoS scenario and observed a significant improvement on prediction of QoS values and service ranking.

We have considered the impact of using different number of neighbouring datasets on our prediction results. We found that smaller number of similar datasets can provide good results. We have evaluated the influence of the meta-attributes considered and found that these meta-attributes affect the service behaviour in different ways.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

We have addressed the problem of QoS based service selection and ranking for data analytic services. Several meta-attributes can influence the service behavior thereby affecting the service selection and ranking results. The previous techniques cannot provide accurate ranking results as they do not consider these dataset characteristics. In this thesis, we have proposed a methodology that can be used to predict the QoS attributes of data processing services according to the nature of the dataset and then rank them.

We distinguished between two kinds of data-dependent QoS attributes – "per dataset data-dependent QoS attributes" and "per service data-dependent QoS attributes". The "per dataset data-dependent QoS attributes" can include latency and accuracy since these values vary for each service for a specific dataset. For "per service QoS attributes" such as reliability, the value is computed for the service based on all the datasets that have been processed by the service.

Our contributions are as listed below:

- We provided a method for prediction of the "per dataset data-dependent QoS attributes" values of a service based on a collaborative approach used in recommendation systems.
- The services are ranked based on the predicted QoS results using a utility based approach. This way, our system is capable of recommending services for new datasets.

- We have considered accuracy in the QoS-based service selection process. Accuracy is seldom used as an attribute in QoS-based service selection work, to the best of our knowledge. While in many web-services such as weather forecasting, e-mailing services, accuracy may not be a concern; it is certainly a crucial quality aspect of data analytic services.

We considered 5 meta-attributes for our similarity computation: dataset size, number of dimensions, percentage of missing data, type of data and the data distribution pattern. With the help of our experiments, we were able to establish the relevance of using these attributes in our meta-learning system.

We found that our prediction and ranking results are significantly better than the traditional QoS ranking approach which is to take average QoS values for ranking purpose. This proves the importance of using a meta-learning approach for this problem. We also have provided the user an option to automatically select the similarity threshold of similar datasets which can provide accurate prediction results.

5.2 Future Work

There are several aspects we can consider for our future work. We would like to work with more data-dependent QoS attributes such as response time and throughput. We can also incorporate non data-dependent QoS attributes such as availability, accessibility, integrity etc. to provide a comprehensive QoS based selection and ranking system.

We can consider different QoS based ranking methodologies as well. We could employ ranking methods that incorporate user constraints and system constraints, based on which the service matchmaking procedure optimizes the results. We would also like to test different preference values for the QoS attributes based on the importance of the attributes which can be assigned according to the context and application of the service.

We can work with different meta-learners such as the data mining algorithms used in recent research towards the ranking of machine learning algorithms that include the kNN, SVM or neural network algorithms. We plan to test this approach on real-datasets such as the datasets from the UCI repository as well.

We are also interested to work with a wider range of meta-attribute values and would like to test other meta-attributes such as noise and work with multivariate datasets. Furthermore, we would like to study the interrelationship between the meta-attributes such as the effect of noise on numeric datasets versus nominal datasets. We can work with different weights for the meta-attributes used in the prediction of different QoS attributes based on the level of impact of the meta-attribute.

Finally, we can apply our selection mechanism for other applications, such as different data mining services like data classification services or regression services. We could experiment with such a selection mechanism for streaming data classification or clustering to handle big data. We could also delve into recommendation of other services that deal with data such as data preprocessing, data compression and data conversion services etc.

REFERENCES

[1] M. P. Papazoglou, P. Traverso, S. Dustdar and F. Leymann, "Service oriented computing: State of the art and research challenges," *Computer*, vol. 40, pp. 38-45, 2007.

[2] M. P. Papazoglou, "Service-oriented computing: Concepts, characteristics and directions," in *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE),* 2003, pp. 3-12.

[3] M. Crasso, A. Zunino and M. Campo., "A Survey of Approaches to Web Service Discovery in Service-Oriented Architectures," *Journal of Database Management,* vol. 22, pp. 102-132, 2011.

[4] M. P. Papazoglou and W. Van Den Heuvel, "Service oriented architectures: approaches, technologies and research issues," *The VLDB Journal,* vol. 16, pp. 389-415, 2007.

[5] A. Birukou, E. Blanzieri, V. D'Andrea, P. Giorgini and N. Kokash, "Improving web service discovery with usage data," *Software, IEEE,* vol. 24, pp. 47-54, 2007.

[6] Y. Liu, A. H. Ngu and L. Z. Zeng, "QoS computation and policing in dynamic web service selection," in *Proceedings of the 13th International World Wide Web Conference,* 2004, pp. 66-73.

[7] E. Al-Masri and Q. H. Mahmoud, "Qos-based discovery and ranking of web services," in *Proceedings of 16th International Conference on Computer Communications and Networks (ICCCN)*, 2007, pp. 529-534.

[8] J. Yan and J. Piao, "Towards QoS-based web services discovery," in *Service-Oriented Computing–ICSOC 2008 Workshops,* 2009, pp. 200-210.

[9] D. A. Menascé and V. Dubey, "Utility-based QoS brokering in service oriented architectures," in *IEEE International Conference on Web Services (ICWS),* 2007, pp. 422-430.

[10] S. Lamparter, A. Ankolekar, R. Studer and S. Grimm, "Preference-based selection of highly configurable web services," in *Proceedings of the 16th International Conference on World Wide Web,* 2007, pp. 1013-1022.

[11] C. Herssens, I. J. Jureta and S. Faulkner, "Dealing with quality tradeoffs during service selection," in *International Conference On Autonomic Computing (ICAC),* 2008, pp. 77-86.

[12] V. X. Tran, H. Tsuji and R. Masuda, "A new QoS ontology and its QoS-based ranking algorithm for Web services," *Simulation Modelling Practice and Theory,* vol. 17, pp. 1378-1398, 2009.

[13] D. Skoutas, D. Sacharidis, A. Simitsis, V. Kantere and T. Sellis, "Top-k dominant web services under multi-criteria matching," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology,* 2009, pp. 898-909.

[14] Q. Yu and A. Bouguettaya, "Computing service skyline from uncertain QoWS," *IEEE Transactions on Services Computing,* vol. 3, pp. 16-29, 2010.

[15] A. K. Jain, M. N. Murty and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys (CSUR),* vol. 31, pp. 264-323, 1999.

[16] J. Han and M. Kamber, *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2011.

[17] D. Pelleg and A. W. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters." in *International Conference on Machine Learning (ICML),* 2000, pp. 727-734.

[18] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 2005.

[19] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of Operations Research,* vol. 10, pp. 180-184, 1985.

[20] Y. Yang, X. Guan and J. You, "CLOPE: A fast and effective clustering algorithm for transactional data," in *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD),* 2002, pp. 682-687.

[21] J. Han and M. Kamber, *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2006.

[22] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning,* vol. 2, pp. 139-172, 1987.

[23] M. Ester, H. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Knowledge Discovery and Data Mining (KDD),* 1996, pp. 226-231.

[24] E. Al-Masri and Q. H. Mahmoud. (October 2008). *The QWS Dataset*. Available: http://www.uoguelph.ca/~qmahmoud/qws/#Definitions.

[25] A. Ruiz Cortés, O. Martin-Diaz, A. Durán and M. Toro, "Improving the automatic procurement of web services using constraint programming," *International Journal of Cooperative Information Systems*, vol. 14, pp. 439-467, 2005.

[26] K. Kritikos and D. Plexousakis, "Mixed-integer programming for QoS-based web service matchmaking," *IEEE Transactions on Services Computing*, vol. 2, pp. 122-139, 2009.

[27] Q. Ma, H. Wang, Y. Li, G. Xie and F. Liu, "A semantic QoS-aware discovery framework for web services," in *IEEE International Conference on Web Services (ICWS)*, 2008, pp. 129-136.

[28] H. Wang, C. Lee and T. Ho, "Combining subjective and objective QoS factors for personalized web service selection," *Expert Systems with Applications*, vol. 32, pp. 571-584, 2007.

[29] I. Janciak and P. Brezany, "A reference model for data mining web services," in *Sixth International Conference on Semantics Knowledge and Gerid (SKG)*, 2010, pp. 251-258.

[30] A. Congiusta, D. Talia and P. Trunfio, "Distributed data mining services leveraging WSRF," *Future Generation Computer Systems*, vol. 23, pp. 34-41, 2007.

[31] S. Benkner, I. Brandic, G. Engelbrecht and R. Schmidt, "VGE-a service-oriented grid environment for on-demand supercomputing," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, 2004, pp. 11-18.

[32] M. Reif, F. Shafait and A. Dengel, "Dataset generation for meta-learning," *Poster and Demo Track of the 35th German Conference on Artificial Intelligence*, pp. 69-73, 2012.

[33] D. G. Ferrari and L. N. de Castro, "Clustering algorithm recommendation: A meta-learning approach," in *Swarm, Evolutionary, and Memetic Computing,* Springer, 2012, pp. 143-150.

[34] C. Lemke, M. Budka and B. Gabrys, "Metalearning: a survey of trends and technologies," *Artificial Intelligence Review,* pp. 1-14, 2013.

[35] K. A. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Computing Surveys (CSUR),* vol. 41, pp. 6, 2008.

[36] R. G. Soares, T. B. Ludermir and F. A. De Carvalho, "An analysis of meta-learning techniques for ranking clustering algorithms applied to artificial data," in *Artificial Neural Networks–ICANN* Springer, 2009, pp. 131-140.

[37] M.C.P de Souto, R. B. C. Prudêncio, R. G. Soares, D. S. de Araujo, I. G. Costa, T. B. Ludermir and A. Schliep, "Ranking and selecting clustering algorithms using a meta-learning approach," in *IEEE International Joint Conference on Neural Networks (IJCNN),* 2008, pp. 3729-3735.

[38] M. Lichman. (2013). *UCI Machine Learning Repository*. Available: http://archive.ics.uci.edu/ml.

[39] P. B. Brazdil, C. Soares and J. P. Da Costa, "Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results," *Machine Learning,* vol. 50, pp. 251-277, 2003.

[40] M. Vukićević, S. Radovanović, M. Milovanović and M. Minović, "Cloud Based Metalearning System for Predictive Modeling of Biomedical Data," *The Scientific World Journal,* vol. 2014, 2014.

[41] Q. Sun and B. Pfahringer, "Pairwise meta-rules for better meta-learning-based algorithm ranking," *Machine Learning,* vol. 93, pp. 141-161, 2013.

[42] H. Blockeel and J. Vanschoren, "Experiment databases: Towards an improved experimental methodology in machine learning," in *Knowledge Discovery in Databases: PKDD* ,Springer, 2007, pp. 6-17.

[43] J. Vanschoren, H. Blockeel, B. Pfahringer and G. Holmes, "Experiment databases," *Machine Learning,* vol. 87, pp. 127-158, 2012.

[44] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter,* vol. 11, pp. 10-18, 2009.

[45] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering,* vol. 17, pp. 734-749, 2005.

[46] T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *ACM SIGMOD Record,* 1996, pp. 103-114.

[47] S. Rahman, "A test-bed for QoS-based data analytic service selection in the cloud," 2015.

[48] C. Sammut and G. I. Webb, "Encyclopedia of machine learning," in Springer Science & Business Media, 2011, pp. 652.