CHAPTER 3
LITERATURE SURVEY

3.1 Preamble
Recently, Web service discovery has been a hot topic. Accordingly, many approaches to discover WS have been built. In order to enable a fluid reading of the analyzed approaches they have been categorized in four groups such as those approaches that base on classic information retrieval techniques, those that base on QoS information, those that base on semantics, and those that based on data mining.

This chapter includes the surveyed works related to Information Retrieval techniques, various QoS driven approaches and WS recommender system that use CF or CBF mechanisms, data mining approach for WS selection/recommendation, some promising ranking methods originally presented in different fields, but are being used in the Web service selection systems and server log analysis in Information Retrieval (IR) systems. Also, this chapter includes the evaluation report of the surveyed works. Finally, this chapter concludes by putting emphasis on open problems and future research opportunities.

3.2 Web Service Discovery Approaches
Industrial and academic researchers have proposed many approaches to assist providers deliver the most relevant WS to particular user’s needs. Some approaches analyzed Web service descriptions [36, 34, 37, 61], some studied the QoS of services [39, 40, 62], while others used semantic concepts to discover services [35, 41, 42].

Different methods in information retrieval, data mining and artificial intelligence domains were experimented such as CF [34, 39, 40, 35], clustering [36, 37, 63], divide and conquer [37] and so on. Different types of applications such as similarity search engines [36, 34, 37, 64] and web service RS [65, 66] were also developed. In order to clarify the difference between the existing approaches and identify the need of a Web service RS based on user’s behavior, they are classified into four groups based on the used methodologies, including information retrieval or text-based, QoS-based, semantic-based and past usage-based. The following section includes the details of the solutions in each category.
3.2.1 Information Retrieval-based Discovery

Recently, a number of methods have been developed in IR systems to lessen the problem of discovering relevant services to the well-known problem of finding relevant documents. Under this reduction, a “document” consists of a service description described in WSDL which is having an record in an UDDI registry.

Most IR-based approaches based on linear algebra were found to be appropriate substitutes for correlating similar documents. The basis of such approaches is to represent the documents and queries as vectors and then finding out the most similar vectors. In Vector Space Model (VSM), a vector $\mathbf{v} = (e_0, e_1 \ldots e_n)$ represents a document, whose elements $e_i$ denote the importance of each distinct word $w_i$ for that document and the vectors of similar documents will have similar representations. VSM is a model which represents both the documents and the queries as vectors where the similarity of vectors shows the similarities between the corresponding documents or queries. Similarity of WS is computed based on the similarities of vector elements. The similarities between Web service descriptions are computed by the similarity between the corresponding concepts using Term Frequency-Inverse Document Frequency (TF-IDF) measure.

In IR based approaches, the discoverers have a chance to look up services by providing a natural language description, or a set of keywords. This added ability to declare the queries provides the explorers with a “Google-like” interface for inquiry. One of the main drawbacks of these approaches, on the other hand, is that they depend on publishers’ best practices usage for commenting and naming services, operations and arguments. This is because the underlying matching mechanism of IR-based approaches does syntactic comparison of strings.

Also the IR-based discovery approaches are having rich background as they are inherited from former research on classic document retrieval. Generally, text mining techniques like removing non relevant words (a.k.a. stop-words), bridging synonyms and removing the commoner morphological and inflectional endings from words (a.k.a. stemming), have been recognized as being very suitable to improve the performance of these approaches. Some researchers have used the above-mentioned techniques for finding the syntactic differences at Web service descriptions.

Dong et al. [36] propose a method for computing the similarity between WS by using the
structures of WS which includes name, text, operation descriptions, and input/output description. They also proposed a search engine called Woogle which includes similarity search functionality for WS.

Platzer and Dustdar [34] proposed to match the user’s query string with Web service descriptions. However, they match the query vector directly with document vectors without structuring the input, output and operation vectors or clustering them in different concepts discussed in [36]. They firstly collect Web service descriptions, i.e. WSDL files, from different resources, such as user’s uploading, links from websites, or references from UDDI repositories. Then, each Web service description file is parsed to generate a corresponding vector. Each element in the vector corresponds to a word in the description and its value is the number of time that the word appears in the description. The user’s query string is also represented as a vector. Each term in the vectors is weighted by TF-IDF and the similarity between a query-vector and a document-vector is computed by VSM, i.e. the cosine of the angle between the two vectors. Finally, for each query string, the authors recommend the WS whose descriptions have the highest similarity values with the query string.

In [27], Blake et al. attempted to recommend WS that are relevant to user’s daily routine. Different from [36] and [34], which manipulated the user’s query strings and Web service descriptions, they examined the similarity between the text data that a user was viewing/processing and the operations of a Web service. They firstly capture the text data that a user is working on such as HTML files, Word documents, File systems, messages (ICQ, SOAP, etc.). Then, they extract text strings from the captured data. These extracted strings are compared to the operations of available WS to infer the similarity between them. In their approach, they captured four naming tendencies that software designers/developers used to name a Web service and proposed to apply Levenshtein Distance and Letter Pairing to compute the similarity between two strings. Finally, WS that have the highest similarity values are recommended to the user.

Stroulia and Wang [67], Kokash et al. [68], Lee et al. [69] adapt the VSM to translate Web service descriptions into vectors. The authors also traduce keywords-based queries into vectors. Then, service look up operates by comparing vectors.

Zhuge and Liu [70] propose an approach by complementing syntactic exact matching
techniques by linking terms that semantically include other terms, even when the terms are from different syntactical point of view. To achieve this, the authors propose a powerful definition called flexible matching for similarity calculation between services. This definition enables discoverers to look for identical services which are more specific or more general than their queries. For example, flexible matching considers the distance between two categories in taxonomy or if a requested set of keywords, which denotes operation inputs, is completely or partially met. Another contribution of this discovery approach is that developers find it easy to learn its query language because the syntax and semantics are borrowed from Structured Query Language. This approach uses a relational model of Web service data based on the schema information present in UDDI.

Kokash [71] relates IR-based most relevant algorithms for evaluating the similarity between two Web service descriptions. He reports an assessment of different algorithms using the same collection of WS and finally concludes by mentioning that algorithms based on the classic statistical measure are used to assess how important a word is to a document, namely, TF-IDF, over performed other approaches in most cases.

Liu et al. [72] propose to extract 4 features, i.e., content, context, host name, and service name, from the WSDL document to cluster WS. They take the process of clustering as the preprocessor to discovery, hoping to help in building a search engine to crawl and cluster non-semantic WS.

Elgazzar et al. [73] also propose to extract features from WSDL documents to cluster WS. Different from the work [90], Khalid et al. [63] extracts content, types, messages, ports, and service name from WSDL documents. He proved a better precision value when compared to Liu’s work using different features.

Liu [72] and Elgazzar [73] combine string-based similarity methods such as structure matching with a corpus-based method based on Normalized Google Distance to measure the similarity of Web service features and to cluster them appropriately. However, structure matching may not accurately identify the semantic similarity among terms because of the heterogeneity and independence of service sources. These methods consider terms only at the syntactic level. Further, there can be a loss of the machine-interpretable semantics found in service descriptions when converting data provided in service descriptions into vectors in string
based IR techniques. Moreover, Normalized Google Distance does not take into account the context in which the terms co-occur, and, although the method uses up-to-date knowledge and information from the Internet, it does not encode fine-grained information, leading to low precision in the clustering results.

It is noticed that the works employing only the syntactic information available in the WSDL document are not satisfactory in guiding a user suitable service selection. This is basically because inside a WSDL interface, there may not be enough textual description, which can be used for indexing a service satisfactorily. Thus the retrieved services may include many impertinent services when services are retrieved based on index and this makes difficult the selection of a needed service. The observation of various surveyed IR-based approaches is given in the Table 3.1.

Table 3.1: Limitations of various information retrieval/text-based approaches

<table>
<thead>
<tr>
<th>Authors(s)</th>
<th>Observations / Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dong et al. [36]</td>
<td>• Feature considered are name, text, operation descriptions, and input/output description.</td>
</tr>
<tr>
<td></td>
<td>• Data types are not considered properly and QoS is not considered for searching</td>
</tr>
<tr>
<td>Platzer et al. [34]</td>
<td>• QoS factors not considered</td>
</tr>
<tr>
<td>Kokash [68]</td>
<td>• WordNet is used to expand the WSDL concept descriptions which as a general dictionary does not cover all terms and term meanings in every specific subject</td>
</tr>
<tr>
<td>Liu et al. [69]</td>
<td>• Considered features are content, context, host name, and service name.</td>
</tr>
<tr>
<td></td>
<td>• The service context and service host name features provides limited assistance in the clustering process.</td>
</tr>
<tr>
<td></td>
<td>• Providers generally provide different Web services on their own website.</td>
</tr>
<tr>
<td></td>
<td>• Mining the near web pages or the consideration of host name provides no meaning of the WS, which is not the case in UDDI.</td>
</tr>
<tr>
<td></td>
<td>• In addition, some WS does not use of the &lt;documentation&gt; element in the WSDL document meaning that there is inadequate information for the content feature.</td>
</tr>
<tr>
<td>Elgazzar et al. [73]</td>
<td>• Used corpus based normalized Google distance method to measure the difference between two WSDL documents.</td>
</tr>
<tr>
<td></td>
<td>• Knowledge-based methods lack up-to-date information and the methods have the problem of shortage of high-quality ontologies.</td>
</tr>
<tr>
<td></td>
<td>• They do not encode fine-grained information and that leads to reduce the precision of clustering results.</td>
</tr>
<tr>
<td></td>
<td>• Quality of the retrieved results is not evaluated.</td>
</tr>
</tbody>
</table>
3.2.2 Semantic-based Discovery

To face the insufficient syntactic side of Web service, several approaches were proposed for dealing with semantic WS. In this field of works, the relevant Web service selection depends on a common knowledge between the provider and the consumer. This common knowledge is embodied in ontology. This kind of works attempts to solve the problem of selecting WS, by considering a unique ontology. If several ontologies are considered, mapping between ontology need be carried out, which is yet another challenge.

Machine interpretable description of Web service is another feature that text-based discovery approaches doesn’t support [13]. Members of the Semantic Web community recommend that by annotating services with ontologies, discoverers can access to an unambiguous shared definition of each part of a WS (e.g., input, output, operation, etc.). For example, if the output message of an operation is named “temp”, a discoverer might not be able to precisely infer what it refers to. Otherwise, if this output is related with a concept which defines the current temperature of a region, then the meaning can be understood by the discoverer [14].

Also, semantic matching algorithms are supported by ontologies. Semantic Web service is one the description of which consists of a machine interpretable definition of its constituent parts. Differing from syntactic matching, semantic matching allows distinguishing between syntactically equivalent terms, but semantically different. This is needed by software agents which attempts for automatic discovery of services i.e., without any intervention of human discoverers.

However, the high levels of automatic discovery can only be achieved at the cost of engaging effort on correctly specifying ontologies, managing them and annotating services. The three main efforts for defining the metamodel for describing a Web service are Web Ontology Language for Services [OWL-S] [13], Web Service Modeling Ontology [WSMO] [74] and WSDL-S [75]. OWL-S [13] offers a framework which can be used for describing both the functions and advertisements for WS by using OWL. OWL is a W3C recommendation for describing the semantic relationships of a domain. Though, OWL-S includes three sub-ontologies, namely, Service Profile, Process Model and Grounding, the Service Profile sub-ontology is directly related to service discovery as it defines the functionalities provided by a
service. Generally, this sub-ontology lets publishers to annotate preconditions, inputs, outputs, effects of Web service operations and also few non-functional attributes.

OWL is designed to represent machine interpretable content on the Web. OWL is based on RDF-S, a structured language based on XML. RDF-S extends the Resource Definition Framework (RDF) with a set of predefined types, high level constructors (e.g., class, subclassOf, property) and range and domain constraints over properties. Above RDF-S, OWL defines more facilities (e.g., inverseOf, equivalentClass, sameAs, symmetry) for expressing meaning and semantics than XML, RDF, and RDF-S.

A Web service recommender system based on semantic matching and rating prediction was proposed by Manikrao et al. [12]. They proposed to describe WS as ontologies using DARPA Agent Markup Language (DAML) (latterly Web Ontology Language-OWL). Then, they matched all semantic attributes of WS and considered that two WS are similar if their individual semantic attributes matching is greater than given thresholds. The proposed system can predict the rating of a user to a WS based on her previous ratings on other similar WS. In their proposition, two services are more similar if their average ratings are less different.

Paliwal et al. [42] implement a Web service discovery engine that uses the information contained in the plain WSDL files, Latent Semantic Analysis (LSA), and several external domain ontologies. The main idea is to build and train an LSA classifier created on features mined from the WSDL files and use it derive a measure of similarity between the service request and a set of WSDL files by projecting the description vectors and the request vector into the LSA vector space and comparing them using cosine similarity. The domain ontologies are used to enhance the original service request with associated concepts that are determined as relevant.

Yamine [76] uses a semantic registry for semantic WS, which are well-found with an exploitation language for assisting semantic-based discovery. Nabil et al. [77] proposes an approach based on service ontologies and semantic indexations. They propose a persistent architecture centered on ontology-based database to store and index the various services, as well as their compositions. The prototype implements semantic concepts for service and workflow. This enables storing, retrieving, reusing existing services and workflows, and building new ones incrementally.

Xia Wang et al. [78] advise a QoS-based semantic Web service selection mechanism.
They use the Web Service Modeling Ontology (WSMO) to describe a QoS model, including specific quality metrics, value attributes, and their respective measurements. They take into consideration user quality requirements.

Benatallah et al. [79] recommend a matchmaking algorithm. It takes as input a service request and ontology, and then finds a set of services whose descriptions contain as much common information with the query as possible, and as little extra information with the query as possible.

Nayak et al. [80] propose a Web service discovery approach with additional semantics and clustering. They acquired benefits of the OWL-S ontology and WordNet dictionary for enhancing the description with semantics. Each of the extracted terms from the service documents was expanded to enhance its semantics by using WordNet to identify synonyms. They used the Jaccard coefficient in computing the service similarity.

Semantic-based approaches that exploit the latent semantics hidden in Web service descriptions [34, 27, 42] can generate recommendations without creating ontologies or asking any effort from users. In this work, the latent semantics hidden in the WSDL document is examined.

The authors in [34] also applied SVD and Latent Semantic Indexing (LSI) and provided experiments on precision and recall metrics. However, instead of filtering WS in two phases like Ma et al. [37], they applied VSM and LSI directly on the corpus of Web service documents. Concretely, they presented Web service documents in a corpus as a matrix of documents and terms. Then, they decomposed the document-term matrix to an approximate matrix using SVD. Finally, they matched the query terms and the document rows in the decomposed matrix in order to find the closest documents to a given query.

Instead of describing WS as ontologies like [12, 77, 78, 79], Ma et al. [37] explored the semantic concepts hidden in the Web service descriptions. They proposed a two-phase approach that applies the divide and conquer methodology and singular value decomposition technique. In the first phase, the collections of Web service descriptions are divided into a set of smaller clusters by using the divide and conquer approach. In this phase, VSM and TF-IDF are applied to syntactically match the query and Web service descriptions in clusters to identify the most relevant cluster to the query. In the second phase, the SVD technique is applied on the selected
cluster to decompose the document vectors and present them in a reduced space. Similarities between the query vector and the decomposed document vectors are computed by the cosine similarity between these vectors. Based on these similarities, the relevant services to the query are identified.

Tables 3.2 and 3.3 summarize the observations of limited surveyed semantic approaches.

Table 3.2: Limitations of various semantic approaches

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Standard/External Source</th>
<th>Observations/Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manikrao et al. [12]</td>
<td>DAML (latterly OWL)</td>
<td>• Creating and publishing ontology annotated content is time-consuming and error prone task as it needs to be done by domain experts using questionable editing tools</td>
</tr>
<tr>
<td>Yamine Aït Ameur [76]</td>
<td>OWL-S</td>
<td>• If several ontologies were used, ontology mapping must be carried out, which is yet another challenge</td>
</tr>
<tr>
<td>Nabil Belaid [77]</td>
<td>Ontology</td>
<td></td>
</tr>
<tr>
<td>Wang et al. [78]</td>
<td>WSMO</td>
<td></td>
</tr>
<tr>
<td>Benatallah et al. [79]</td>
<td>Ontology</td>
<td></td>
</tr>
<tr>
<td>Nayak and Lee [80]</td>
<td>OWL-S and WordNet</td>
<td>• WordNet do not include WSDL terms that are not proper English words</td>
</tr>
</tbody>
</table>

Table 3.3: Limitations of various LSA-based approaches

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Observations/Limitations</th>
</tr>
</thead>
</table>
| Paliwal et al [42]         | • LSI directly on the corpus of WS documents  
  • Uses LSA and several external domain ontologies and suffers from performance issue |
| Jiangang Ma et al. [37]    | • Only three features namely, service name, operation and port types are considered.  
  • WSDL documents are clustered using bisecting K-Means algorithm |

3.2.3 QoS-based Discovery

From the viewpoint of service users, selecting an appropriate service is a serious step to build a dependable software system. Commonly, service selection is generally associated with
considering QoS attributes. Accordingly, QoS prediction for WS and ranking WS has caused widespread attention in the field of service computing. Many QoS driven approaches have been proposed for Web service search, Web service recommender system, Web service selection and so on. The various works proposed by different researchers in presented below.

In [81], Zhang et al. proposed a search engine that took into account both functionality and QoS for service ranking. The functionality is described by the Web service description, such as operation, input and output. The QoS is non-functionality features, such as price and response time. A user’s query includes both functionality and QoS requests (Fig. 4.4). In the QoS query, ‘Constraint’ is a vector of queried values corresponding to QoS attributes. The ‘Weight’ vector allows user to specify the weight of each QoS attribute in her query. The authors computed both the functionality matching and the QoS matching. Parameter $\lambda \in [0, 1]$ is used to balance the bias of these matching.

Different from [81], Zibin Zheng et al. [82] took into account only QoS values of WS to build a Web service recommender system. They aim at predicting the probability that a user uses a Web service $i$ in order to recommend her the WS that have the highest probabilities. To do so, they built a user-item matrix, where the rows present the user IDs, the columns present the Web service IDs and each entry of this matrix is a vector of QoS values observed by the corresponding user on the corresponding Web service. Based on this matrix, they compute the similarity between WS and users using Pearson Correlation Coefficient. The probability that a user $a$ will use a Web service $i$ is formulated based on the computed similarities. Two QoS metrics that are taken into account for their experiments are the round-trip-time and the failure-rate of the requests to each WS.

Pradnya Khutade et al. [83] propose a two-step method namely, matchmaking and selection of Web service for web service selection. First in sub-process, WSDL documents satisfying QoS requirements are matched and the result is a list of Web service fulfilling user’s QoS request. In second sub-process, result returned by first sub-process is sorted using weights of QoS metrics of the service consumer.

Yuliang Shi et al. [84] propose a method for Web service selection which provides the users the approximate QoS values, and helps in discovering the optimal WS. Initially, the users are clustered using location and network condition and then considering the QoS historical data.
A time-aware personalized QoS prediction framework for WS was proposed by Yilei Zhang et al. [85]. From past user’s usage scenario, data is collected and a feature model is built. Personalized WS selection based on previous user’s QoS prediction.

A Web service selection method based on ranking of QoS using associative classification was proposed by Molood Makhlughian et al. [86] where Web service selection is done by ranking based on semantic matching and ranking based on QoS parameters considering service consumer’s preferences. This work involves three processes namely, preprocessing of QoS attributes, service selection by local classification and ranking of WS based on functional aspects.

Raj et al. [87] propose Web service selection based on QoS constraints where the QoS manager acts as an agent for service providers and service requester for publication and retrieval of required WS. The proposed algorithm in this work is used for optimizing the QoS attributes and for ranking. For each query, the request is processed by the QoS manager and returns the matched WS useful for setting QoS preferences.

Mohammad et al. [88] propose as hybrid solution in order to combine the global optimization with local selection techniques. An optimal agent-based architecture was proposed by Rajendran et al. [89] for dynamic Web service discovery using QoS attributes. The architecture has four components namely, service publisher, verifier and certifier, retrieval agent, quality analyzer and WS Storage. The agent service is used for accessing service registry (UDDI). The architecture helps clients in Web service selection process which based on QoS parameters.

Gang Ye et al. [90] propose a QoS based model for WS discovery by introducing QoS broker. The proposed model finds WS with real-time, fair and authentic QoS values by using the proposed monitoring and valuation mechanism.

Mohamad Mehdi et al. [91] propose probabilistic approach for Web service selection based on QoS attributes. This work involves a probabilistic method for forecasting the quality of a Web service based on the assessment of previous experiences (ratings) by its consumers. QoS ratings are denoted by using different statistical distributions such as Multinomial Dirichlet, Multinomial Generalized Dirichlet, and Multinomial Beta-Liouville. Bayesian inference method is employed to estimate the parameters of the mentioned distributions, which presents reliable
WS to service consumer.

Lina Yao et al. [92] propose an innovative approach by combining CF with CBF features which recommends WS dynamically by satisfying users’ interest. The proposed work involves a hybrid approach of both CF and CBF recommender systems and proved that the proposed hybrid system is outperforming the latter two recommendation system in terms of recommendation accuracy.

Qi Yu [93] propose a service selection scheme by providing an automatic method for assessing QoS values for unknown service providers and thereby provides a reliable Web service matching service requester’s query. Relational clustering based model is used for effectively addressing the data scarcity issue. The proposed automation model shows a better accuracy with respect to prediction of QoS values.

Yali L et al. [94] propose a hybrid method considering user-based and item-based CF algorithms by using as improvement method for similarity calculation by adopting Pearson Correlation Coefficient for measuring the similarity between two users or two services.

Zibin Zheng et al. [95] propose a CF recommendation method for forecasting QoS values for WS by using the historical usage experience of service requester. Initially, a user-collaborative mechanism is proposed for collecting historical WS QoS information from different service requester. Finally, using the collected QoS data, a novel CF recommendation is proposed for forecasting QoS values. A prototype named WSRec is implemented using the proposed methods and the observed result showed as improvement on prediction accuracy than traditional approaches.

An effective personalized CF method for WS recommendation is proposed by Yechun Jiang et al. [96] by proposing a key component for computation of similarity measurement of WS. Compared with the Pearson Correlation method for similarity measurement, the proposed method considers the personalized influence of services when computing similarity measurement between users. Also an effective personalized hybrid CF technique is developed by integrating personalized user based and personalized item based algorithms. They proved that the proposed method has a better accuracy of recommendation of WS considerably.

Huifeng Sun et al. [97] propose a normal recovery CF method for personalized Web service recommendation with an innovative similarity measurement technique. A personalized
context aware QoS prediction method was proposed by Qi Xie et al. [98] for WS recommendations using Slope One method and the proposed work considers context, which is an important factor in both RS and QoS forecasting. Experimental results proved that the recommended approach offers better QoS forecasting.

CF systems use user feedback, such as ratings, to reflect users’ opinions or experiences on performance or quality, which would be a major factor to be considered in a RS. The explicit user feedback systems, such as reputation based or community feedback based systems, usually involve human efforts to provide feedback or ratings. However, not all users are willing to provide feedback or ratings after each usage [20], and furthermore, the user ratings might not be accurate or trustworthy [99].

Although the past usage data are important resource exposing users’ interests, there are still few approaches that take these data into account for WS discovery. Birukou et al. [66] proposed to utilize the past usage data for a Web service recommender system. They defined an ‘Implicit Culture’ concept as a relation between a set and a group of agents such that the elements of the set behave according to the group’s culture. Based on this concept, they consider that if a user has a similar request with other members of the community, he will be recommended operations that were used by those members. Therefore, they record the usage of users together with their requests. And if they receive a new request, they will recommend the requester Web service operations that are used by other users who have similar requests. The similarity between service requests is computed based on VSM, TF-IDF and WordNet-based semantic similarity. However, the author did not process the past usage data. They used them in their proposed rule-based theory for generating recommendations. In addition, the similarity between requests is computed using text-based approaches. Hence, they ignore the correlation between users and WS in the past usage data, which can infer the users’ interests. Moreover, they can meet the shortcomings of text-based approaches while matching users’ requests.

L. Sha [100] introduced WSSM-Q by using a QoS management module to evaluate the quality of WS in compared to the user's requirements. In this work, they considered only four non-functional attributes: availability, price, latency and performance. Their work consists of two levels: QoS information collecting and, QoS information processing. In the QoS information collecting stage, they use a monitor facility based on a handler provided by the JAX RPC and
JAX WS specifications. The quality information of the requested Web service from the time a user sends a request until he receives the result is collected and sent to the management module. In the QoS information processing level, all the quality data are calculated and stored in a QoS repository.

Y. Liu and H. He [101] proposed a vector-based ranking algorithm to measure the goodness of the matched services and make recommendations to users based on their requirements. They modeled all the published services in a vector. They also modeled the query QoS requirements and their related weights in an n-dimension binary vector. The weights are required to be assigned by consumer. Then they calculated the distance between the query vector and the published. In their framework, all services are sorted based on their distance scores. The framework solves the problem by returning the final ranked list based on a minimum score.

Al-Masri et al. [102] suggest that a perfect trusted broker which allows publishers to provide only QoS values which cannot be predicted or forecast (e.g., cost per invocation), whereas other QoS values are computed automatically. To achieve this, the proposed broker monitors service invocations and collects QoS values about WS. This approach also presents a function for ranking WS based on their QoS characteristics which had been gathered by the broker. The function works out a matrix for representing a Web service in a row and each QoS parameter in a column. The QoS values are normalized as QoS parameters differ in units and magnitude. Another matrix is computed by dividing each column by the maximum normalized value that occurs within it. Then a vector of weights indicating the importance level assigned to each QoS parameter is considered. The larger the weight the more important its associated QoS parameter is to the client. The values of these weights range from 0 to 1 and all weights must add up to 1. Finally, these weights are introduced into the matrix as factors and all values are recalculated. The row that maximizes the sum of its weighted parameters represents the first ranked WS, and so on.

In addition, Y. Shi [103] offered a linear regression prediction algorithm clustering user in respect to location and network condition considering QoS values. It is normal to discover that the distance between users has an effect on prediction precision, however, which is not easily measured in practice.

Wang et al. [104] propose a new CF algorithm based on Slope One algorithm combining
uncertain neighbors with Slope One method. Zhu et al. [105] proposes a personalized recommendation algorithm which combines Slope One method and user based CF which utilizes Slope One method to forecast the missing QoS ratings of the user item matrix. Then it utilizes the user based CF to produce the recommendation. The experiments were made on a common data set using different filtering algorithms.

The limitations/observations of the limited surveyed QoS-based approaches are included in Table 3.4 and Table 3.5 presents the limitations and evaluations of collaborative filtering-based web service selection methods.

Table 3.4: Limitations of various QoS-based approaches

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Observations/Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pradnya et al. [83]</td>
<td>• WSDL language is unreliable to track QoS attribute values.</td>
</tr>
<tr>
<td></td>
<td>• Matchmaker algorithm is not capable to take correct decision using QoS parameters and hence they deliver incorrect WS with QoS constraints.</td>
</tr>
<tr>
<td>Yuliang et al. [84]</td>
<td>• Clustering based on users QoS historical statistics/ratings given by users can’t be taken into account because a service provider, acting as malicious users can rate its target service provider with poor ratings.</td>
</tr>
<tr>
<td></td>
<td>• Only few QoS parameters are considered.</td>
</tr>
<tr>
<td>Zhang et al. [81]</td>
<td>• QoS features considered are price and response time.</td>
</tr>
<tr>
<td></td>
<td>• Does not consider the user’s QoS requirements in ranking process.</td>
</tr>
<tr>
<td></td>
<td>• Only few QoS parameters are considered.</td>
</tr>
<tr>
<td>Molood et al. [86]</td>
<td>• Implementation of utility value dependence for each Web service is not a reliable approach because the utility value threshold has to be set high enough for reliable classification of WS based on CBA algorithm.</td>
</tr>
<tr>
<td>Raj et al. [87]</td>
<td>• Only 5 QoS attributes namely availability, throughput, response time and cost are considered.</td>
</tr>
<tr>
<td></td>
<td>• Constant QoS ranks are assigned, which leads to poor search results - needs to be changed for each type of WS.</td>
</tr>
<tr>
<td></td>
<td>• User’s QoS requirements are not considered in ranking process.</td>
</tr>
</tbody>
</table>
Only response time, throughput, price and availability parameters are considered.

Web service agent is not a consistent approach as agents are not a reliable component as it can be easily overridden by malicious users.

Relying on third party, broker is unreliable as it is possible for any competitive service provider to act as malicious user and cause harness to QoS broker.

Only few QoS parameters are considered.

### 3.2.4 Data Mining-based Discovery

With the advancement of the World Wide Web, Web service discovery has become an area of research. The process of service discovery is a three-step process including parsing of the user’s query, matching the description against a catalogue and finding a combination of the matched constraints that satisfy the cost constraints of the query. The problem of computing cost-constrained collections of WS can be linked to the hyper graph traversal problem.

In order to apply the graph traversal problem in Web service discovery, pre-processing methods can be applied to make the results optimal. Edge pruning, horizontal and vertical partitioning are being used to reduce the size of the dataset before applying the depth first enumeration or breadth first enumeration. This helps to achieve a more accurate and relevant result satisfying all of the constraints.

The skip graph based methods may be used to find the WS of interest. The skip graph is a randomized and balanced tree data structure on which redundant connectivity and multiple handles have been added. Skip graph based methods support complex searches with locality preserving features of skip graph [106]. In order to find the maximum benefit, WSDL-S have been used so that input, output, pre-condition and effect can be considered for service matching. The Vector Space Model is being used to compute the similarity between the user query and WSDL-S document or between WSDL-S documents.

Integrating data-mining techniques in Web service discovery can lead to an improved and accurate WS discovery. The lack of semantic description in WS leads to inadequate number of search results [107]. Many of the results provided as an outcome to user query partially qualify to user’s interest. Semantic Web service clustering utilizes the
<table>
<thead>
<tr>
<th>Author(s)</th>
<th>User-based CF</th>
<th>Item-based CF</th>
<th>Hybrid CF</th>
<th>Model-based CF</th>
<th>Approach</th>
<th>Related Attributes</th>
<th>Algorithm</th>
<th>Observations/Limitations</th>
</tr>
</thead>
</table>
| Mohamad Mehdi et al. [91]| X            | X             | X         |                | Bayesian Network  | (RT T A R)        | Bayesian Network Classifier                      | • Inflexible Quality of predictions  
• Synonyms Problem                                            |
| Sheng et al. [92]         | X            | X             | √ (CF and Content-Based) | X             | X                 | (RT T)            | Pearson Correlation Coefficient and Content based | • Malicious user ratings                                      |
| Qi Yu et al. [93]         | √            | X             | X         | X             | X                 | (RT T A R)        | Cosine Similarity                                  | • Data Sparsity  
• Scalability  
• Doesn’t generalize data  
• Gray sheep problem                                      |
| Yali LI et al. [94]       | √            | √             | √         | X             | X                 | (RT T A R)        | Pearson Correlation Coefficient                   | • Malicious user ratings                                      |
| Zheng et al. [95]         | X            | √             | X         | X             | X                 | (RT T)            | Correlation Similarity                            | • Data Sparsity  
• Scalability  
• Malicious user ratings  
• Doesn’t generalize data  
• Gray sheep problem                                      |
| Yechun Jiang et al. [96]  | √            | √             | √         | X             | X                 | (RT T A R)        | Pearson Correlation Coefficient                   | • Malicious user ratings                                      |
| Huifeng et al. [97]       | X            | √             | X         | X             | X                 | (RT T)            | Pearson Correlation Coefficient                   | • Malicious user ratings                                      |
| Chen et al. [98]          | X            | X             | X         |                | Slope One Method  | (RT T)            | Slope One Method                                  | • Time-consuming  
• Cold Start problem  
• Gray sheep problem                                          |

P - Price  
A - Availability  
RT - Response Time  
T – Throughput  
R – Reliability  
√ - Indicates presence  
X - Indicates absence
semantic representation for grouping similar WS to improve the service discovery. In order to enhance the semantics, the WSDL documents are annotated using OWL-S so that features of service profile, process model and grounding can be utilized. To calculate the similarity between WS for clustering, [107] have considered the Jaccard’s coefficient. In order to group similar WS, a clustering technique has been proposed. The accumulative similarity co-efficient is calculated considering the WS description as well as the OWL-S features. Web service search engine Woogle also utilizes clustering and association mining to find similarities between WS based on common user queries.

Clustering is a method of grouping data items using similarity measure. It is a subjective process such as the same set of data items often needs to be partitioned differently for different applications which make it difficult [108]. Clustering is an important step in the process of data analysis, which partitions objects into groups, known as clusters. The objects in the same group are similar and that in different groups are dissimilar. In order to perform clustering a number of clustering algorithms like hierarchical agglomerative, K-means etc. can be used [109].

Hierarchical Clustering starts at each point regarded as a cluster, and recursively combines pairs of clusters until all points are part of one hierarchical constructed cluster. The resulting cluster can be represented by a dendrogram, which are in the form of [110]. The dendrogram can be easily broken at selected links to obtain clusters of desired cardinality or radius. This is of great interest to a number of application domains.

In most cases, the agglomerative algorithms are used in situations where, initially, each object is assigned to its own cluster and then pairs of clusters are repeatedly merged until the entire tree is formed. The heuristics of this algorithm is that it starts with n clusters, each consisting of exactly one instance and it computes the distance matrix between instances. It then searches for the most similar pair of clusters, reducing the number of clusters by one, due to the pair merger. The entries are updated after this. It is different from others such as K-Means, because the classical K-means algorithm provides a flat clustering of points in a vector-space [111].

Other data mining techniques can also be used for recommending the WS of interest to the user. Predictive Mining can be used for recommending service to the end user based on
user’s search criteria. Classification techniques like decision tree can be used to build rules on service description [107]. Web service recommendation can be based on the principles of association mining. The Web server logs containing records of interaction between WS and users can be utilized to find the similar usage patterns. Based on similar usage pattern, the similar WS can be grouped so that while searching for similar WS the result set contains more number of similar services [107].

3.3 Rank Aggregation Algorithms

Many Web service discovery and ranking methods are inspired by the effective techniques developed in the database community. Rank-based aggregation technique proposed by Aslam and Montague [112] is one of the methods used in this context. In this model, first the services are ranked in different lists based on each individual attribute, and then the algorithm combines different ranked lists to compute the final ranked list.

Rank aggregation problem is the problem of how to aggregate m ranked lists generated by n sources. According to the latest researches, there are two types of rank-based aggregation methods as shown in Fig. 3.5 namely, supervised rank aggregation technique which relies on the training data and unsupervised rank aggregation method with no need of the training data. Unsupervised rank aggregation technique is categorized in two groups such as positional methods and majoritarian techniques.

![Figure 3.1: Rank aggregation methods](image)

Positional methods generate the final ranked list based on the combination of all ranking scores gained by summing all the positional values of each element in each ranked list. The most
common method in regard to the rank aggregation method is the Linear Score Combination method in which the scores of items are aggregated by some operators such as weighted sum to compute the final ranked list.

Another important algorithm in this context is referred as Borda Fuse proposed by Bartell et al. [113]. It is considered an effective algorithm to rank a set of data points. The algorithm was introduced to solve the voting problem. Supposing that there are \( n \) voters to rank a fixed set of \( m \) data points, the user will get \( n \) ranked lists including \( m \) data points. For each individual ranked list, the top ranked item receives \( m \) points, the second candidate receives points, and so on. The last item in the list receives 1 point. The final score for each candidate is calculated based on the summation of all \( n \) points assigned to the candidate. The item with the most points will be ranked the best. It is a very simple procedure, which has been proved to be effective.

Another positional algorithm that can be named in this area is Median-Rank aggregation method introduced by Fagin et al. [114], in which the documents are ranked based on their median ranks. Given \( m \) data points and \( n \) lists of values assigned to each point, the ranking process works as follows: first, all \( m \) documents are sorted according to their values in the lists, thus \( n \) ranked lists will be returned; then, the final rank list is computed as the median of the positional values of each element in the rank lists. The final ranked list is generated by sorting the median value of each element. This method is not able to support ties.

Footrule optimal rank aggregation method [115, 116] is another type of positional rank aggregation methods. The ranking prototype is used to minimize the Spearman Footrule distance from the input rankings. Based on this theory, for any given two rank lists \( l_i \) and \( l_j \) in a set of \( n \) lists, the Spearman Footrule distance is computed by using Eqn. 3.1.

\[
SFD (l_i , l_j ) = \sum_{i=1}^{n} \left| r_{i}^{li} - r_{i}^{lj} \right|
\]  

where \( r_{i} \) is the ranking position of candidate \( i \) in each related list. The lower value of Spearman Footrule Distance shows the more similarity between the two mentioned lists.

Majoritarian rank aggregation approaches are another type of unsupervised rank aggregation methods. In this type of algorithms every item is compared with another candidate [117]. The method consists of repetitive steps. First they considered a list of all candidates and then each item in the list is compared with the next one. The winner stays in the list, but the loser will be removed from the list. The comparison steps are repeated until there is no other item in
the list to be compared. This method suffers from low speed, as the number of comparisons gets larger, when the number of items in the dataset increases.

Condorcet-Fuse method proposed by Montague and Aslam [118] is one of the voting models based on Majoritarian rank aggregation technique. This model is also based on pair-wise comparisons. Each candidate is compared to all other items in the dataset in terms of all QoS attributes. The model works based on a theory that the item which wins in majority of pair-wise comparisons, will appear in the final ranked list. However this method suffers from high computation time, and lack of capability to deal with ties, i.e. the situation that it is not feasible to choose a winner from a comparison contest.

In this research work, positional methods such as Linear Score Combination and Borda Fuse are considered as the baseline methods as they are widely used and efficient.

3.4 Skyline Operation

There is another type of matchmaking and ranking algorithms based on Skyline query concept which is a dominant topic in the database field. The skyline operation was introduced by Kossman et al. [119] to solve maximum vector problems. The model calculates and filters the desired points relevant to a query and returns all possible solutions among a large set of data points in a given domain. Suppose that a client is seeking cheapest hotels closer to the shopping centers. It is difficult to choose a hotel between all possible options, as closer hotels to the shopping center tend to be more expensive. According to the skyline operation, the desired hotels are those not worse than the others in both dimensions. The ultimate set of desired hotels is called as skyline points. Skyline points are collection of services which are not dominated by other services. A service $S_i$ dominates service $S_j$ if it is better than $S_j$ in at least one attribute and not worse than the service in other attributes. Skyline points assist consumers to select their desired service easier based on their preferences. From the graphical point of view, the name of skyline was selected for the computation of result set.

In context of the skyline query field, Papaïdas et al. [120] introduced a progressive algorithm which relies on Branch and Bound Skyline using a nearest-neighbor search method. On a given set of points, this model computes the skyline points based on their distances to a query point in an ascending order. In this work, they first indexed the data by applying an R-tree technique to reduce the computation cost by decreasing the number of pair-wise comparisons.
Then they computed the dominance relationship between each two services. They argued that in their framework any pre-computation functions would not be required. Branch and Bound Skyline is widely used in multi-criteria optimization problem.

To extend the Skyline query model to relational databases, [121] presented a new algorithm called Sort-Filter-Skyline (SFS) model. They implemented their model based on a sorting technique. According to their theory, all data points are sorted by using a sorting technique and considering a monotone function. In other words, SFS sorts all candidates that maximize the scoring function in an ascending order. After sorting the data, the services which dominate the other services over most attributes will appear in upper positions. Thus the number of pairwise comparisons will decrease. Any service with the best score over the monotone function will appear in the skyline list. This method is used extensively and is a fundamental structure for methodologies invented later. Hence, in this research work this model is also considered as a baseline for comparison purpose.

Skoutas et al. [122] proposed a new algorithm for ranking and clustering WS according to the dominance concept. Their model supports multi criteria matching without combining matching scores of each individual parameter. The model combines Top-K queries and skyline operation. A threshold is also considered in this work to compute the probability of being in skyline for each service. The algorithm is composed of three steps as given below and their model also reflects the trade-offs between matched parameters.

- Step 1: Select the services that their probability to be in the skyline above the threshold.
- Step 2: Select k representatives from the list returned in the previous step.
- Step 3: Form the clusters by assigning the other services to their related cluster.

**3.5 Log Data Used in Web Search Personalization**

There are many researches on web usage mining for web personalization [123]. Web usage data records users’ interaction with the web and could usually be stored in a log file which is located at either client side or server side. Client side log can collect usage data from an individual user who often interacts with multiple web sites, whereas server side log can collect usage data from multiple users who access the web site hosted on the server. Hence, client side log is often used in content-based RS, and server side log can be used in both content-based and collaborative filtering based recommender systems.
There are different types of usage data, including the browsing history, the click-through data, time spent on a page, actions applied on a page (such as printing and saving), the transaction history, and so on. Search engines normally rank the level of relevance of a web document based on the frequency of the query keywords appeared in the content of the document. However, for short and ambiguous queries, search engine performance will be deteriorated. The click-through data was used in [124] to improve the performance of the search engine under such ambiguous conditions. The click-through data was extracted from a large amount of log data collected by search engine servers. The log normally contains search queries, followed by the uniform resource locator of the web page clicked by the user. The user’s click stream reflects the user’s opinion about the page relevance. An Iterative Algorithm was proposed to compute the page similarity as well as the query similarity based on the concept that the web pages visited by similar queries are similar; and search queries visiting similar web pages are similar. The experiment showed a big improvement on the search performance.

The effectiveness of using implicit user feedbacks in web search ranking has been studied in [125]. This work modeled the user search behavior as a combination of “background” information and “relevance” information where “background” information represents noise information. User actions for each search result were represented as a vector of features. It could include any type of user interaction that is collected by search engine logs and these data are categorized into click-through features, browsing features, and query-text features. Then, a ranker was trained to discover feature values that are relevant on search results to produce a trained user behavior model which was used to help the ranking process. It used a simple merge algorithm which computed the merge score of a document based on its rank from the implicit feedback-based ranks and the original ranks. The result showed a significant improvement in the final performance.

3.7 Chapter Summary
Discovering WS in the Internet has been widely recognized as a very difficult task. Therefore, different approaches to alleviate service discovery have been proposed in the literature. Earlier sections have analyzed some of the most associated approaches related to this research work. Unfortunately, the approaches discussed do not cope with a subset of the problems that are
essential to truly enable the discovery of services on environments that manifest an incessantly increasing number of services. The problems are:

- There is a lack of effective techniques for discovering services that, at the same time, do not impose heavy costs on publishers. Semantic-based methods show a higher retrieval efficiency surpassing the efficiency attained by syntactic-based approaches. On the other hand, the semantic styles makes publishers to put extra effort while describing services by means of semantic meta-data as they refer to particulars of services using machine understandable languages. But when search engines are used, more effort is required to describe the goods accurately in order to get the better retrieval effectiveness. This situation makes the service publication hard to gain retrieval effectiveness. However, in order to achieve better retrieval effectiveness, lightweight and effective approaches to service discovery should spring.

- As Web service technologies are widely adopted, it is expected to have massive growth in the number of published services. The success of Web service and SOC depends on the capabilities of service discovery systems to cope with ever increasing number of publicly available services. In recent times, various researchers have attempted to discover the ability of several service registries in order to deal the growing number of publicly available services. Evidently, the important concern in this direction is the study of scalability of the infrastructure which is supporting the discovery process. But it is not the only direction to be explored. Discoverers are annoyed with the increased number of WS in search results. There is a lack of search space reduction techniques, which allow discoverers to promptly obtain proper WS, in spite of the huge number of alternatives. Therefore, more effort is required for improving the scalability of discovery systems for without ignoring its retrieval efficiency.

- It is important to note that, most of the earlier research works on QoS prediction methods used Pearson-based similarity method. This method is good in prediction but costs much computation time and also loses performance for the very sparse data set. Thus there is a need for a new approach to improve the prediction accuracy which will finally improve the quality of recommendations.
• Most of the reviewed models for ranking WS are reasonable work. However, most of the methods suffers from low speed, as the number of comparisons gets larger, when the number of items in the dataset increases and they mostly either ignore the role of users' requirement or their methods require users to compute the importance degree of each parameter. On the other hand they put more load on users. Consumers tend to use applications which run fast without involving them in the computations. Also they only considered a limited number and a few types of attributes (mostly numeric type), while in reality there are various types of data. Hence these issues are to be addressed by developing a simple, but effective method which considers user requirement as an important factor in the ranking process without imposing any further load on consumers considering different number and types of data.

• CF systems use user feedback, such as ratings, to reflect users’ opinions or experiences on performance or quality, which would be a major factor to be considered in a recommender system. The feedback may be explicit or implicit. In the explicit user feedback system, not all users are willing to provide feedback or ratings after each usage, and furthermore, the user ratings might not be accurate or trustworthy. As the explicit user feedbacks are often impractical or unable to completely reflect users’ true opinions or experiences, this issue to be addressed.

• Inherent problem with the CF systems, namely, the cold start problem needs also to be addressed.

    This research works addresses the above-mentioned limitations with respect to the existing works for Web service recommendation and the proposed approaches to overcome the limitations are discussed in detail in the forthcoming chapters.