CHAPTER - 2
PRELIMINARIES AND BACKGROUND

2.1 Preamble
Service-Oriented Architecture (SOA) [21] is a design paradigm that suggests the construction of applications by the composition of existing software modules that are called services. In this vision, Web services are considered an important realization of this application design paradigm. In this research work, a solution for Web service selection is proposed, for enabling WS consumption. This is mainly essential for the realization of Web Service-Oriented Architectures.

This chapter gives the key basics and definitions surrounding the Web service technology. Also presents the basics of recommender system and highlights the potential benefits they offer. Next the popular ways in which the current literature classifies recommendation approaches are reviewed. This provides both the foundation and the context for the contributions in this thesis. Finally, the chapter concludes by listing the limitations related to Web services, their standards and recommendation approaches.

2.2 Overview of Web Service
In the literature, several definitions for Web services are found. A definition from [22] that introduces a WS is as follows:

"A WS is a platform-independent, loosely coupled, self-contained, programmable Web-enabled application that can be described, published, discovered, coordinated, and configured using eXtensible Markup Language (XML) artifacts (open standards) for the purpose of developing distributed interoperable applications."

Indeed, a Web service is an application that is accessible through the Internet. It is self-contained, meaning that it functions independently without external support. It is self-describing through an XML-based public interface containing a URL for service binding, and a specification of the functionality which other software systems can discover. Then, the other software systems work together with the WS as given in its definition by using XML-based messages as mentioned in Internet protocols [23, 24].

The objectives of Web service which can be invoked from remote location are providing a precise functionality or a group of functionalities over a network. The term Web service
defines a uniform method to integrate Web based applications using the Web Service Description Language (WSDL) [25] which describes the services, Simple Object Access Protocol (SOAP) which is used for data transfer, XML which is used for describing the data in a structured way using custom defined tags and Universal Description, Discovery and Integration (UDDI) [26] which is used for listing what services are available and they are open standards working on Internet protocol.

Dealing with WS involves three actors, a provider, a registry, and a consumer, as illustrated in Fig. 2.1. This figure represents the SOA triangle. It illustrates the interaction between the three actors, as well as the used standards. A service provider publishes his service description at a UDDI registry using a publishing interface. The published interfaces and binding data of the recorded services are well-defined in the WSDL.

A registry organizes the published services and provides a query interface that enables a service consumer to search for a needed service, and obtain its provider’s location information. A service consumer then can interact with a service provider through the SOAP protocol. The three previous actors perform three fundamental WS actions, each of which is related to a standard.

![Figure 2.1: The SOA triangle](image)

**2.2.1 Describing a Web Service**

Describing a Web service is the action using which, a service can be represented (exposed and advertised) to the external world. A functionality of a service is represented to the external world by an abstract interface described using the standard WSDL.
2.2.1.1 Functional Properties

WSDL is an XML-based language that contains several elements to describe a Web service’s functionality. It consists of two parts, an abstract part and a concrete one. The abstract part defines the service interface, i.e. the operations together with their parameters and data types. The data types are described as XML schema elements. The concrete part describes how to bind to and invoke the service concerned, by specifying the binding protocol together with the service’s endpoint. For example, the WSDL interface for a math service, called MathService is shown in Fig 2.2.

```xml
<wsdl:definition>
  <! The types section --->
  <wsdl:types> … </wsdl:types>
</wsdl:definition>

<wsdl:definition>
  <! The messages section --->
  <wsdl:message name="AddSoapIn"> …. </wsdl:message>
  <wsdl:message name="AddSoapOut"> …. </wsdl:message>
  <wsdl:message name="AddHttpGetIn"> …. </wsdl:message>
  <wsdl:message name="AddHttpGetOut"> …. </wsdl:message>
</wsdl:definition>

<wsdl:definition>
  <! The portTypes section --->
  <wsdl:portType name="MathServiceSoap"> …. </wsdl:portType>
  <wsdl:portType name="MathServiceHttpGet"> …. </wsdl:portType>
</wsdl:definition>

<wsdl:definition>
  <! The Bindings section --->
  <wsdl:binding name=" MathServiceSoap" type="tns:MathServiceSoap"> ….
  </wsdl:binding>
  <wsdl:binding name=" MathServiceHttpGet" type="tns:MathServiceHttpGet"> ….
  </wsdl:binding>
</wsdl:definition>

<wsdl:definition>
  <! The service section --->
  <wsdl:service name="mathService"> …. </wsdl:service>
</wsdl:definition>

Figure 2.2: The WSDL interface of the MathService

It has a root definition element <wsdl:definitions> which is structured into five sections, namely, service section, bindings section, port-types section, messages section and types sections and they are discussed below.

**Service section:** The <wsdl:service> tag specifies the service name, together with its endpoint location for each of the provided interface (portType). For example, in Fig. 2.3, it is
noticed that the MathService has two interfaces "MathServiceSoap" and "MathServiceHttpGet", for which it specifies two distinct endpoints.

![The service section --->

```xml
<wsdl:service name="MathService">
  <wsdl:port name="MathServiceSoap" binding="tns:MathServiceSoap">
    <soap:address location="....."/>
  </wsdl:port>
  <wsdl:port name="MathServiceHttpGet" binding="tns:MathServiceHttpGet">
    <soap:address location="....."/>
  </wsdl:port>
</wsdl:service>
```

Figure 2.3: The service part inside the MathService WSDL interface

**Bindings section:** <wsdl:binding> tag specifies the protocol for each binding, together with the provided operations. For example, in Fig. 2.4, the MathService provides two bindings "MathServiceSoap" and "MathServiceHttpGet" corresponding to the two interfaces it provides. For the "MathServiceSoap" interface, the binding specifies that the used protocol is SOAP using <soap:binding>.

**PortTypes section:** <wsdl:portType> tag specifies the interfaces (portTypes) that the service provides. For example, in Fig. 2.5, the MathService provides two portTypes "MathServiceSoap" and "MathServiceHttpGet". The "Math-ServiceSoap" also provides an operation called "Add", which takes an input message called "AddSoapIn" and returns an output message "AddSoapOut".

![The bindings section --->

```xml
<wsdl:binding name="MathServiceSoap" binding="tns:MathServiceSoap">
  <soap:binding transport=".”/> 
  <wsdl:operation name="Add">
    <wsdl:operation soapAction="…””/>
    <wsdl:input>…</wsdl:input>
    <wsdl:output>…</wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Figure 2.4: The binding part inside the MathService WSDL interface
<wsdl:portType name="MathServiceSoap">
  <wsdl:operation name="Add">
    <wsdl:input message="tns:AddSoapIn" />
    <wsdl:output message="tns:AddSoapOut" />
  </wsdl:operation>
</wsdl:portType>

Figure 2.5: The portType part inside the MathService WSDL interface

Messages section: <message> tag includes the input/output parameters for each operation. For example, in Fig. 2.6, the input/output messages for the “Add” operation are shown. The input message "AddSoapIn" has one parameter of type "AddRequest", and the output message "AddSoapOut" that has one parameter of type "AddResponse".

Types sections: <wsdl:types> tag includes the parameter types of each operation inside the WSDL file are specified as an XML schema element. For example, the MathService defines a schema of two elements "AddRequest" and "AddResponse", as shown in Fig. 2.7. They represent the parameters types for "AddSoapIn" and "AddSoapOut" messages, respectively. The "AddRequest" element is a sequence of two elements of type float, while the "AddResponse" has only one element of type float.

<wsdl:message name="AddSoapIn">
  <wsdl:part name="parameters" element="tns:AddRequest" />
</wsdl:message>
<wsdl:message name="AddSoapOut">
  <wsdl:part name="parameters" element="tns:AddResponse" />
</wsdl:message>

Figure 2.6: The messages defined for the "Add" operation inside the MathService

<! The messages section --->
<wsdl:message name="AddSoapIn">
  <wsdl:part name="parameters" element="tns:AddRequest" />
</wsdl:message>
<wsdl:message name="AddSoapOut">
  <wsdl:part name="parameters" element="tns:AddResponse" />
</wsdl:message>

<! The types section … >
<wsdl:types>
  <s:schema ... >
    <s:element name="AddRequest">
      <s:complexType>
      ...
    </s:complexType>
  </s:element>
</s:schema>

</wsdl:types>

<! The portTypes section …>
Figure 2.7: The WSDL types defined for the MathService

2.2.1.2 Non-Functional Properties

Non-functional properties of a Web service are mainly used to model QoS attributes. QoS denotes the capability of the Web service to respond to expected calls performing at a level corresponding to the mutual expectations of both its provider and its consumer [27]. The QoS is influenced by the Internet’s dynamic and unpredictable nature. Therefore, delivering good QoS is a critical and significant challenge.

QoS attributes such as constant availability, connectivity, and high responsiveness are the key to keeping a service competitive and viable. They represent important criteria to determine the service usability and effectiveness, which has an impact the WS’s popularity. Especially with the growing number of functionally similar WS on the internet, there should be a technique to differentiate between them using a set of well-defined QoS attributes and values. Thus, QoS has become an important part of service description, for a better service selection [28] and composition [29]. Some of the QoS attributes can be calculated by the consumer of a WS (like response time), while other attributes must be calculated and provided by the Web service’s provider such as scalability and security. Below mentioned are some of the main QoS attributes:
- **Availability** indicates the Web service’s presence during a period of time, as a percentage.
- **Throughput** represents the number of WS requests served at a given time period.
- **Latency** is the time between sending a request and receiving the response.
- **Reliability** is the quality aspect of a WS that represents the degree of being capable of maintaining the service and service quality.
- **Response time** is the average time (in milliseconds) that is needed to obtain a response from a WS.
- **Security** represents the provider’s approaches and levels of providing security by confirming the parties involved, encrypting messages and providing access control.

### 2.2.2 Discovering a Web Service

Web service discovering is associated to registries and repositories, where one can search for a needed service. A registry must provide a fairly precise search capability to render the discovery action more efficient and less time-consuming. Initially, Web service registries were supposed to be built upon a standard called UDDI, but discovery mechanisms are not limited to this standard.

UDDI is an XML-based standard that was proposed for WS registries, for allowing providers to publish their services, so that they can be located afterwards by consumers. A UDDI registry is structured into three layers, as illustrated in Fig. 2.8 and the role of each layer is mentioned below.

```
businessEntity: information about the Web service providers
    (name, identifier, category, discoveryURL, …)

businessService: information about the Web Service offered by the provider
    (name, bindingTemplate, category, …)

bindingTemplate: Technical information about a Web service
    (description, category …)

tModel: Description of specifications of services or taxonomies
    (name, description, identifier, category, …)
```

**Figure 2.8: Overview of UDDI data structure**
- businessEntity or white pages, which list the providers and their related information;
- businessService or yellow pages, which provide a classification of services based on standard taxonomies;
- bindingTemplate or green pages, which provide information about the service bindings and their types tModel.

These layers permit searching a UDDI registry according to the three data types: businessEntity, businessService, and tModel. Thus, the possible searching queries are: search for business by (business name, identifier, category, discovery URL); search for service by (service name, category), and search for service type by (service type name, category).

Presently, Web service discovery is not limited to UDDI-based registries. WS can be carried out through either Web service portals (service search engines) or through websites of certain service providers. The previous SOA triangle (Fig. 2.1) can be rather represented as in Fig. 2.9.

![Figure 2.9: The refined SOA triangle](image)

The Web service architecture creates fundamentals for the WS development. Many companies developed and published their WS. A survey of Gartner on 110 companies [30]
showed that 54% are working on WS. In another survey on 2847 executives worldwide in 2007 [31], 80% companies were using or going to use WS and 78% identified that WS are the most important technologies for their business. The growing rate of the number of WS collected by search engines from Oct. 2006 to Oct. 2007 is 286% [32]. However, since 2006, public UDDI registries (SAP, IBM and Microsoft) were withdrawn [33].

The growing quantum of WS and the discontinuing of public UDDI registries make WS more scattered. Many Web service portals, such as XMethods, BindingPoint, WebServiceX.NET, or Web service crawlers, such as Seekda and EmbraceService were developed to allow service providers to continue publishing their service descriptions. They also provide interfaces for users to search and invoke WS. The website skeeda.com which has been launched since 2006 has collected around 28000 WS from 7739 providers. Meanwhile, [9] registry stores more than 8000 WS. Apart from them, users can also discover WS via search engines such as Google, Yahoo, All the Web or Baidu.

The Web service technology has been proven as an efficient means to delivering services to users. They are used worldwide in most of the companies. However, the Web service concept is still not familiar to end users, which still get difficulty in discovering WS. They need advanced mechanisms to assist them to better discover services. This requirement leads to many researches on proposing solutions to enhance Web service discovery. Some approaches analyze Web service descriptions for better matching with the query strings [34, 27, 35], other approaches group WS in clusters [36, 37], some approaches examine the quality of services [38, 39, 40], whereas [41, 42] rely on the semantic descriptions of WS.

2.2.3 Invoking a Web Service

Web service invocation describes a client-server communication and is embodied in the messages swapped among service provider and service consumer. These messages are described by a standard called SOAP and are transmitted over a Hypertext Transfer Protocol (HTTP) connection.

SOAP is an information exchange protocol used in “big” WS. Messages are defined using XML and are transmitted using protocols such as HTTP or Simple Mail Transfer Protocol. The specification has four parts such as description of the SOAP message, the rules to
serialize messages, definition of the protocol binding between SOAP and HTTP, and how to use SOAP for Remote Procedure Call like binding.

The SOAP message consists of an envelope, a header, and a body. The envelope is the root element of the XML document and it is mandatory. The header is an optional element that contains metadata about the contents of the message. It is useful because an application can check its contents to determine if it needs to process the whole body of the message or not. Finally, the body element is a mandatory element that contains all the data that needs to be transmitted.

The data transmitted using SOAP has to be serialized into a format understood by the protocol. SOAP uses the same simple data types found in the XML Schema definition. Complex types can be formed by combining these simple types. It is usually straightforward to transform the data types in most programming languages to this representation. HTTP is the most common protocol used to transport SOAP messages. There are several ways to use SOAP with HTTP. The semantics of SOAP map naturally to the semantics of HTTP.

However, SOAP messages can also be transported using other protocols. Most services available at the time of writing use version 1.1 of the SOAP specification. A newer version of the specification, version 1.2, is also available and it includes new features that allow better interoperability, better protocol independence, and increased performance. However, for the purpose of WS discovery, the difference between protocols is irrelevant.

2.3 Overview of Recommender System

A Recommender System is defined as “any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options” [43]. E-commerce websites usually use RS for providing consumers with information to help them to make decisions on purchases. They can summarize the consumers’ opinions and critiques, provide personalized product information to consumers and suggest products. From a broader view, the RS can help the website to adapt to each consumer and for supporting the customization of the consumer experience. The consumer experience includes the physical products and the presentation of the products, both of which can be customized according to the consumers’ preferences [48].
**Steps of Recommender System**

RS consists of two basic entities such as the user (sometimes also referred to as customer) and the item (also referred to as product) where the user is the one who consumes the RS for providing his judgment on different items and receives recommendations about new items from the system. The working of a RS consists of mainly three different steps namely, input to RS, employing different filtering approaches and finally the output as shown in Fig. 2.10 and they are explained below.

![Diagram of Recommender System Steps](image)

**Figure 2.10: Recommender system steps**

**Step 1 - Input:** RS gets different inputs depending on the kind of the filtering approaches and usually the input goes into one of the categories mentioned below.

- **Ratings:** The ratings depict the view of users on items and they are usually given by the user and follow a specified numerical scale (example: 1-bad to 5-excellent).
- **Demographic data:** It denotes to information such as the age, the gender and the education of the users and this is generally collected explicitly from the user.
- **Content data:** It is based on the textual analysis of documents related to the items rated by the user. The features extracted by this analysis are used as input to the filtering method in order to infer a user profile.

**Step 2 – Employing Different Filtering Algorithms:** The goal of RSs is to generate suggestions about new items or to predict the utility of a specific item for a particular user. In both cases, the process is based on the input provided, which is related to the preferences of that user. Let $m$ be the number of users $U = \{u_1, u_2 \ldots u_m\}$ and $n$ be the number of items $I = \{i_1, i_2 \ldots i_n\}$. Each user $u_i$, where $i = 1, 2, \ldots, m$, has a list of items $I_{ui}$ for which he has expressed his opinion about. User opinions are generally stated in the form of ratings. Specifically, the rating of user $u_i$ for item $i_j$, where $j = 1, 2, \ldots, n$, is denoted by $r_{ij}$. All these available ratings are collected in a $m \times n$ user-item matrix. The different filtering algorithms, namely, collaborative,
content and hybrid, employ various techniques either on the rows, which correspond to ratings of a single user about different items, or on the columns, which correspond to different users’ ratings about a single item. A single user $u_a \in U$ as the target user is chosen and define $N \in I$ as the subset of items for which the target user has not expressed his opinion yet, and as a result, for which the RS should generate suggestions.

**Step 3 - Output:** The output of a RS can be either a prediction or a recommendation.

- A Prediction: It is expressed as a numerical value $r_{aj}$ which represents the anticipated opinion of target user $u_a$ for item $j$. This predicted value should necessarily be within the same numerical scale (example: 1-bad to 5-excellent) as the input referring to the opinions provided initially by target user $u_a$.

- A Recommendation: It is expressed as a list of $N$ items, where $N \leq n$, which the target user is expected to like the most. The usual approach in that case requires this list to include only items that the target user has not already purchased viewed or rated. This form of RS’s output is also known as Top-N recommendation.

RS are data driven. The input data of RS are generally classified into two types: explicit and implicit data. The explicit inputs are intentionally provided by users to inform the RS of their preferences. Examples of these inputs are: users’ ratings of items after using them, users’ profile (e.g., personal demographics), or users’ explicit preference (i.e., the information provided by the user which can help the system discern useful items to the user such as keywords, product categories, item attributes). Instead of asking users to provide their preferences as the explicit inputs, the implicit inputs are obtained by observing users’ behaviour. Visit frequency, visit duration, adding an item to basket and purchasing an item etc. are examples of implicit inputs. Furthermore, they also can indicate the different levels of interest in e-commerce websites [44].

Traditionally, the recommendation problem is formulated based on explicit rating structure by the RS community. In the most common formulations, it is reduced to the problem of estimating the rating of an unknown item to a user [20]. Some examples for recommendation problem are given below.

**Example 2.1 (Recommendation problem depiction)** User1 rates Item1 with a “1” on a 1-to-5 scale, which demonstrates he dislikes Item1. Conversely the rating 5 that User 1 gives to Item 3 shows his strong interest in this item, as shown in Table 2.1 which is named as User -Item rating
matrix. The recommendation problem thus can be defined as estimating the rating that User 1 would give to Item4 that he has not used before based on the pattern of his previous ratings.

Traditionally, RS based on a single criterion, a numerical rating, represents how the user likes an item. Two types of entities, users and items, are therefore used for the recommendation, which make the systems two-dimensional. This is represented by the Eqn. 2.1 which is given as

\[ R: User \times Item \rightarrow Rating \] (2.1)

Table 2.1: User-Item rating matrix

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>?</td>
</tr>
<tr>
<td>User 2</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>User 3</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

These systems try to estimate the utility function of the item based on these two entities, user and items. Thus for each user, the item that maximizes the user’s utility will be chosen (as given in Eqn. 2.2). The utility is usually represented by a rating [22].

\[ \forall u \in U, i^*_u = \arg\max_{i \in I} R(u, i) \] (2.2)

**Example 2.2 (Movie recommendation)** Users are asked to give a rating to the movies they have previously watched to present how much they like the movies. From these ratings, the recommender system predicts ratings for movies not yet watched by the users. Those movies with highest predicted ratings are listed as the recommendation.

**Terminology**
A few terminologies frequently mentioned in RS need to be clarified before moving forward. They are prediction, recommendation, discovery and selection.

RS has two phases such as prediction and recommendation [20]. In the prediction phase, the rating of an item for a specific user is predicted using a utility function using the user’s past historical ratings, or the particular item’s content or the profile of the user etc. After forecasting the ratings of all the candidate items for the user, the recommendation phase employs different approaches to select the most suitable items to support the user’s decision. For instance, the Top-
N items from the maximum utility are recommended to the user [20]. However, both the phases are included in a general recommendation method.

Although the term recommendation has been used interchangeably with discovery by some researchers, they are different from each other. Discovery takes a query that describes the current user interests as its input, while recommendation typically tends not to use an explicit query but rather analyzes the available user profile and context [45]. In terms of functionality, discovery focuses on finding a list of items that meet the specified users’ requirements, while recommendation, and selection, deal with choosing the most suitable items on the returned list from discovery. In other words, discovery is a prerequisite for selection [46]. However, they do share some similarities with the aim of “matching a context (which may or may not include an explicit query) to a collection of information objects”, and their distinction has been blurred due to their extension towards personalization [45].

Furthermore, a large amount of research work that covers discovery also includes sorting of the discovery results. Generally, items tend to be recommended after being discovered. Referring to the difference between items selection and recommendation, such as in service domain, the selection process includes discovery and recommendation in [12]. In other words, their recommendation is positioned as a part of the selection framework. However, in [46], selection is defined as involving mapping a set of items to an item, which in its general form is mapping a set of items to a ranking of the items in that set [46]. The perspective taken in this thesis is aligned with the latter view, since discovery, selection and recommendation are similar to each other and each is related to the rankings of objects.

### 2.3.1 Classification of Recommendation Approaches

The most popular classifications of RS are based on how the recommendations are made, i.e., the recommendation approaches. Therefore, RS are commonly categorized into three such as:

- Collaborative filtering approaches in which the recommendation is based on the items liked before by the other similar people having same tastes and preferences
- Content-based filtering approaches (CBF) where the recommendation is made based on the similar items which are liked before by the user and
- Hybrid filtering (HF) approaches where various combinations of CF and CBF are used for making recommendation.
This work mainly presents and discusses the first three common recommendation approaches. Among them, collaborative filtering and content-based approaches are selected as the baseline approaches, and extended with proposed techniques for a better recommendation performance due to they are the most common used and most successful approaches so far.

Most of the further classifications of RSs extend this popular taxonomy mentioned above. For example, Burke [43] extends this classification and adds another three approaches namely, demographic, utility-based and knowledge-based approaches. Schafer et al. [48] classifies recommender systems based on the inputs (e.g., explicit, implicit), the outputs (e.g., suggestions, reviews, ratings, etc.) of the recommender process, the methods used to generate recommendations and the degree of personalization. Anand and Mobasher [44] classify systems based on the data they utilize (e.g., user versus item information), the learning paradigm used (e.g., memory-based versus model-based), the location of the personalization (e.g., client-side versus server-side) and the process that the interaction takes within a user (e.g., reactive versus proactive).

2.3.1.1 Collaborative Filtering Approach
The CF approach is widely implemented in personalized recommendations. Traditionally, it makes recommendations to a particular user based on the other users who share similar tastes and preferences. Those similar users are referred to the active user’s neighborhood. There are many CF systems both in academia and in industry such as GroupLens, Ringo, Amazon.com. CF recommendation approaches are often classified as being either memory-based (or heuristic-based) or model-based. According to Breese et al. [49], memory based approach predicts considering the whole user database, while model based approach makes use of the user database to estimate, or learn a model for forecasting.

Working of Collaborative Filtering Method
The user-based CF approach utilizes the entire user-item database to generate a prediction or recommendation. The working has been shown in Fig. 2.11 and the following steps explain its functionalities.

- **Data Representation:** It builds a database of preferences for items by users. This step deals with the representation scheme used to model the items that have already been
purchased by a user.

- **Neighbourhood Formation:** For each user, historical information is used to identify the similar users (neighbours) who have shown similar behaviour (e.g., accessed the same type of information, purchased a similar set of items) in the past. This step deals with the neighbourhood formation of each user. Different kinds of neighbourhood formation approaches can be employed.

- **Recommendation Generation:** Once the user’s neighbourhood is formed, algorithms can be used for generating Top-K recommendations. This step is called generation of recommendation.

![Figure 2.11: Block diagram of collaborative filtering](image)

The sales in e-commerce can be increased by using RS by three ways such as converting browsers into buyers, increasing cross-sell by advising other products based on what they have purchased and building customer reliability by creating a value-added relationship between the site and the customer. RS help firms to personalize their process of learning about their consumers’ preferences, and enable the web site to present custom interfaces that match consumer needs. The hope is that consumers may reuse the web site. The more a consumer uses the recommender system, the more personalized it can be.

By recommending to the consumers what they want, the relationship between them and the web site will be strengthened and their loyalty to the web site will increase as consumers will normally visit the site frequently which recommends things fitting their requirements best [48].

**A) Memory-based Collaborative Filtering Approach**

One of the earliest works in CF is GroupLens RS [50]. GroupLens is built to help people find articles in the huge stream of available articles based on the opinions of other people. Users are asked to assign ratings ranged from 1 to 5 based on how much they like the article, with 5 highest and 1 lowest. The correlation between the ratings of any two users determines how similar they are to each other. The calculated coefficients are used as the relative weights in
predictions based on the guidance that the ratings of the users who agreed to a user in the past play a more important role and the ratings of the users who disagreed play a less important role in estimating the rating for this user. In general, under the assumption that people who had similar preferences in the past tend to like similar things in the future, predictions on how well a user will like new articles can be made when the ratings of the other users who have rated same articles to this user are known [50].

Systems such as GroupLens use memory-based CF approach. The predictions are made on some incomplete information about the active user and some weighted information from the user database. In other words, the predictions are generated from the whole collection of previously items rated by the users [20]. To provide useful recommendations, the following two points are crucial.

(a) How to measure the similarity of users so as to provide the neighborhood information for the active user and
(b) How to predict the active user’s subjective evaluation of a new item based on the ratings from the other users.

Memory-based CF algorithms use the full or a section of the user-item database to produce a prediction. The neighborhood-based CF algorithm, a predominant memory-based CF algorithm, uses the following steps.

- Calculating the similarity or weight, \( w_{i,j} \) reflecting distance, correlation, or weight between two users or two items \( i \) and \( j \),
- Produce a prediction for the active user by taking the weighted average of all the ratings of the user or item on a certain item or user, or using a simple weighted average [24].

When generating Top-K recommendation, \( k \) most similar users or items (nearest neighbors) is found by computing the similarities, then the neighbors are aggregated to get the Top-K most frequent items as the recommendation.

**Similarity Computation**
Similarity computation between items or users is a fundamental step in memory-based CF algorithms. For item-based CF algorithms, the basic idea of the similarity computation between item \( i \) and item \( j \) is first to work on the users who have rated both of these items and then to
apply a similarity computation to determine the similarity \( w_{i,j} \), between the two co-rated items of the users [24]. For a user-based CF algorithm, first the similarity \( w_{uv} \) between the user \( u \) and \( v \) is calculated considering the users who have both rated the same items. The different methods to compute similarity or weight between users or items are discussed below.

**Correlation-based Similarity**

In this case, similarity \( \text{sim}(u, v) \) among two users \( u \) and \( v \), or \( \text{sim}(i, j) \) between two items \( i \) and \( j \) is achieved by computing the Pearson correlation or other correlation-based similarity methods. Pearson correlation measures the degree to which two variables are directly related with each other and the Pearson correlation-based CF algorithm is a representative CF algorithm, and is widely used in the CF research community.

In the case of user-based algorithm, the Pearson correlation between user \( u \) and \( v \) is computed using the Eqn. 2.3.

\[
\text{sim}(u, v) = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in I_{uv}} (r_{v,i} - \bar{r}_v)^2}}
\]  

(2.3)

where \( I_{uv} \) is the set contains all the items co-rated by user \( u \) and user \( v \). It is the intermediate results for calculating the “closest neighbors” of user \( u \), i.e. \( I_{uv} = \{i \in I \mid r_{u,i} \neq \emptyset \text{ and } r_{v,i} \neq \emptyset\} \) [20].

In the item-based algorithm, \( u \) is considered as the set of user who has rated both items \( i \) and \( j \) and the Pearson correlation is computed using Eqn. 2.4.

\[
\text{sim}(i, j) = \frac{\sum_{u \in I_{ij}} (r_{u,i} - \bar{r}_i)(r_{v,i} - \bar{r}_i)}{\sqrt{\sum_{u \in I_{ij}} (r_{u,i} - \bar{r}_i)^2 \sum_{i \in I_{uv}} (r_{v,i} - \bar{r}_i)^2}}
\]  

(2.4)

where \( r_{u,i} \) is the rating of user \( u \) on item \( i \), \( \bar{r}_i \) is the average rating of the \( i^{th} \) item by those users. Certain variants of item-based and user-based Pearson correlations are found in [45].

Other correlation-based similarities are constrained Pearson correlation, a variation of Pearson correlation that uses midpoint instead of mean rate; Spearman rank correlation, similar to Pearson correlation, except that the ratings are ranks; and Kendall’s \( \tau \) correlation, similar to the Spearman rank correlation, but instead of using ranks themselves, only the relative ranks are used to calculate the correlation [51, 52].
Generally the number of users in the calculation of similarity is considered as the neighborhood size of the active user, and similarity based CF is considered as neighborhood based CF.

**Vector Cosine-based Similarity**

Two documents are measured for similarity by considering each document as a vector of word frequencies and computing the cosine of the angle formed by the frequency vectors [45]. This approach can be considered in CF, which uses users or items instead of documents and ratings instead of word frequencies.

If $R$ is the $m \times n$ user-item matrix, then the similarity between two items, $i$ and $j$, is defined as the cosine of the $n$ dimensional vectors corresponding to the $i^{th}$ and $j^{th}$ column of matrix $R$.

Vector cosine similarity between items $i$ and $j$ is given by Eqn. 2.5.

$$Sim(i, j) = \cos(i, j) = \frac{\mathbf{i} \cdot \mathbf{j}}{||i|| \cdot ||j||}$$

(2.5)

where “$\cdot$” denotes the dot-product of the two vectors. To get the desired similarity computation, for $n$ items, an $n \times n$ similarity matrix is computed [51]. For example, if the vector $\mathbf{A} = \{x_1, y_1\}$, vector $\mathbf{B} = \{x_2, y_2\}$, the vector cosine similarity between $\mathbf{A}$ and $\mathbf{B}$ is given by Eqn. 2.6.

$$Sim_{AB} = \cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{||A|| \cdot ||B||} = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \sqrt{x_2^2 + y_2^2}}$$

(2.6)

In an actual situation, different users may use different rating scales, which the vector cosine similarity cannot take into account. To address this drawback, adjusted cosine similarity is used by subtracting the corresponding user average from each co-rated pair. The Adjusted cosine similarity is similar to Pearson correlation. In fact, Pearson correlation executes cosine similarity with normalized user’s ratings according to his rating manners. So, negative values may occur in the case of Pearson correlation, supposing there are n-point rating scale.

The calculation of all user similarities is one of the factors that can affect the efficiency of recommender systems. To save the calculating time and be practical in real time, user similarities can be calculated in advance using a common method and recalculated once in a while as the deviations in the peer networks is not affected in a short time [20].
Once the active user and all other users’ similarities are calculated, the forecasting or prediction is made based on the other users. The predicted rating of the active user \( u \) for item \( i \), denoted as \( P_{u,i} \), is usually computed as an aggregate of the ratings of the same item provided by the other users in the neighborhood of user \( u \). Some examples of aggregation function from literature [20] are given by Eqns. 2.7, 2.8 and 2.9 respectively.

\[
(a) \quad P_{u,i} = \frac{1}{N} \sum_{v \in V} r_{vi} \tag{2.7}
\]
\[
(b) \quad P_{u,i} = \frac{1}{N} \sum_{v \in V} \text{sim}(u,v) \cdot r_{vi} \tag{2.8}
\]
\[
(c) \quad P_{u,i} = \overline{r_u} \sum_{v \in V} \text{sim}(u,v) \cdot (r_{vi}, \overline{r_v}) \tag{2.9}
\]

where \( V \) indicates the set of \( N \) most similar users to active user \( u \) having rated the item \( i \) as well, \( v \) is any one from the other users, \( r_{vi} \) presents the rating of user \( v \) has given to item \( i \), \( \overline{r_v} \) is defined as the average ratings of user \( v \) has provided based on all the items previously rated, and \( \text{sim}(u,v) \) is the similarity value amongst user \( u \) and \( v \), while \( k \) serves as a normalizing factor, and is normally selected as given Eqn. 2.10.

\[
k = \frac{1}{\sum_{v \in U} \text{sim}(u,v)} \tag{2.10}
\]

The prediction is defined by simply aggregating the ratings of the other users as shown in Eqn. 2.7 However, Eqns. 2.8 and 2.9 are used more often, in which user similarity is used as a weight, i.e., the more similar user \( u \) to user \( v \) is, the more weight rating will play in predicting \( P_{u,i} \). As stated in [20], the introduction of a heuristic artifact \( \text{sim}(u,v) \) enable differentiating between levels of user similarity, while simplifying estimation procedure, as long as a normalizing factor \( k \) is used to normalize the calculations.

Though Pearson method CF achieves better forecasting or prediction accuracy, the performance is not so acceptable when the data matrix is sparse and also costs high computational overhead for measuring the similarity values between users or items. Fortunately, one more well-known method called Slope One has been proved to show improved prediction accuracy when the data matrix is sparse and also shown that the prediction activity is performed with less computing cost. The advantages of Slope One method are:

- Easy to implement and maintain
- Updateable on the fly
- Efficient at query time
• Expect little from first visitors
• Accurate within reason

The Slope One schemes consider information from other users who have rated the same item and from the other items rated by the same user. Also, the schemes depend on data points included neither in the user array nor in the item array, but are still significant information for predicting rating. Much of the strength of the approach comes from data that is not factored in. Specifically, only ratings made by the users who have rated some common item with the predictee user and only those ratings of items that the predictee user has also rated are considered for predicting unknown values under Slope One schemes.

Formally, for any two given valuation arrays $v_i$ and $w_i$ where $i = 1, \ldots, n$, searching for the best predictor is of the form $f(x) = x + b$ to predict $w$ from $v$ by reducing

$$\sum_{i=1}^{n} (v_i + b - w_i)^2$$

(2.11)

Setting the derivative to zero, the value of $b$ is computed using the Eqn. 2.12.

$$b = \sum_{i=1}^{n} \frac{w_i - v_i}{n}$$

(2.12)

In other words, the constant $b$ is chosen to be the average difference between the two arrays. This result motivates the following scheme.

Given a training set $c$, and any two items $j$ and $i$ with ratings $u_j$ and $u_i$ respectively in some user evaluation $u$ (annotated as $u \in S_{i,j}(\chi)$), the average deviation of item $i$ with respect to item $j$ is computed using the Eqn. 2.13.

$$dev_{i,j} = \sum_{e \in S_{i,j}(\chi)} \frac{u_j - u_i}{\text{card}(S_{i,j}(\chi))}$$

(2.13)

Note that any user evaluation $u$ not containing both $u_j$ and $u_i$ is not included in the summation. The symmetric matrix defined by $dev_{i,j}$ can be computed once and updated quickly when new data is entered.

Given that $dev_{i,j} + u_i$ is a prediction for $u_j$ given $u_i$, a reasonable predictor might be the average of all such predictions is given by the Eqn. 2.14.

$$P(u)_j = \frac{1}{\text{card}(R_j)} \sum_{i \in R_j} (dev_{i,j} + u_i)$$

(2.14)

where $R_j = \{ i | i \in S(u), i \neq j, \text{card}(S_{i,j}(\chi)) > 0 \}$ is the collection of all related items. But for a denser data set where ratings exists for almost all pairs of items, that is, where
\( \text{card} \left( S_{i,j}(\chi) \right) > 0 \) almost all \( i, j \), most of the time \( R_j = S(u) \) for \( j \neq S(u) \) and \( R_j = S(u) - \{j\} \) when \( j \in S(u) \).

Since \( \bar{u} = \sum_{l \in S(u)} \frac{u_i}{\text{card}(S(u))} \approx \sum_{l \in R_j \frac{u_i}{\text{card}(R_j)}} \) for most \( j \), the simplified prediction for the Slope One scheme is formulated as Eqn. 2.15.

\[
P(u)_j = \bar{u} + \frac{1}{\text{card}(R_j)} \sum_{l \in R_j} d_{\text{ev}_{i,j}} \tag{2.15}
\]

All the above aggregation functions are computed based on the similarity between users, which are memory-based and also called user-based CF approach (user-to-user CF). Both practical experience and research have reported that memory-based approach such as nearest-neighbor approach has excellent performance in terms of accuracy [53].

**Top-K Recommendation**

Top-K recommendation recommends a set of K top-ranked items which are of interest to a certain user. For example, when a returning customer logs into his http://amazon.com account, he may be recommended a list of books or products that may be of his interest. Top-K recommendation techniques investigate the user-item matrix to determine relations among different users or items and the recommendations are made based on user-item matrix. Model based on association rule mining can be used to make Top-K recommendations.

**User Based Algorithm for Top-K Recommendation**

This type of recommendation algorithm primarily identifies the k most similar users to the active user using the Pearson correlation or vector-space model [50, 54] where each user is treated as a vector in the \( m \)-dimensional item space and the vectors are used for computing similarities between the active user and other users. After the \( k \) most similar users are identified, their corresponding rows in the user-item matrix are combined to recognize a set of items, purchased by the group together with their frequency. With the set, user-based CF techniques then recommend the Top-K most frequent items which the active user has not purchased. User-based Top-K recommendation algorithms have limits related with scalability and real-time performance [54].

**Item-Based Algorithm for Top-K Recommendation**

This type of algorithm was developed to overcome the scalability problem of user-based
recommendation algorithms. In this algorithm, the $k$ most similar items for each item are found first based on the similarity values. Then a set $C$ consisting of candidates of recommended items is formed by taking the union of the $k$ most similar items and removing each of the items in the set, $U$, which has been already purchased by the user. Finally, similarities between each item of the set $C$ and the set $U$ are calculated. The resulting set of the items in $C$, sorted in decreasing order of the similarity, are the recommended item-based Top-K list [54]. One difficulty in this method is when the joint distribution of a set of items is different from the distributions of the individual items in the set; the above schemes can potentially produce suboptimal recommendations. To overcome this, Deshpande and Karypis [55] developed higher order item-based Top-K recommendation algorithms using all combinations of items for a particular size to determine the item sets which are to be recommended to a user.

**B) Model-based Collaborative Filtering Approach**

The growth of machine learning models, data mining algorithms allows the system to study on how to recognize complex patterns based on the training data and then to make smart predictions for the CF tasks for test data or real-world data. Model-based CF algorithms such as Bayesian models, clustering models, and dependency networks have been examined for solving the inadequacies of memory-based CF algorithms [49, 56]. Usually, classification algorithms can be used as CF models if the user ratings are categorical and regression models and Singular Vector Decomposition (SVD) methods are used for numerical ratings. This context is out of focus as this research works focuses on memory based CF.

**2.3.1.2 Content-based Filtering Approach**

The CBF approach assumes that users tend to like similar items to those they have liked in the past. It is built on the analysis of items previously rated by users, and makes recommendations through matching between item characteristics/profiles and user profiles, especially for those textual items, such as documents, web sites, news and books etc. This approach can be used to predict the unseen items that appear as being potential interesting to a user [44].

When these two pieces of information are available such as a description of the item characteristics, and a user profile that shows the (past) interests of this user, the recommendation task can be accomplished by matching the users past interests with the items descriptions. Only those items having a high degree of similarity to the preferences of the user are recommended.
The content-based method is rooted in information retrieval and information filtering research. But it includes additional information from users compared to the traditional information retrieval approaches [20]. The information about user consists of information about the preferences, users’ tastes and requirements. It can be provided explicitly by the users themselves through providing an interface allowing the users to input their preferences or presentations of their interests. It is also collected implicitly from the user’s history such as whether the user has viewed or purchased the item, or the history of the queries typed by the user.

The content representation (also called item profile) of this approach mainly refers the descriptions of the item characteristics. It is often represented by a list of features/attributes for the item, which are also the keywords that characterize an item [16]. For a book recommender, the content could use the author, price, genre, or anything else that describes the item, and store the information into database. One of the most famous content-based systems, namely, Fab system [47] recommends web pages by representing web page content with the most important 100 words.

Suppose $UserProfile(u)$ presents the taste and preferences of user $u$, and shows the features that describe item $s$. The recommendation task transforms to calculating the utility function using the equation (2.16).

$$U(u,s) = score(UserProfile(u), Content(s))$$  \hspace{1cm} (2.16)

In the content-based approach, both user profile and item profile can be presented by keywords. $UserProfile(u)$ can be well-defined as a weight vector consisting of keywords $(w_{1u}, w_{2u}, w_{3u}, \ldots, w_{nu})$ where each weight $w_{iu}$ denotes the importance of keyword $k_i$ to user $u$. $Content(s)$ is defined as $(w_{1s}, w_{2s}, w_{3s}, \ldots, w_{ns})$ and the total number of keywords as $k$.

It seems intuitive that different keywords may have different importance in identifying the items. For instance, a word that appears more often is better suited for characterizing the item. At the same time, it is also possible that some keywords which do not appear very often are very helpful in discriminating among items. Heavier weight should be given to those important words.

There are different ways of calculating the vectors of keyword weights, such as Rocchio algorithm (i.e., it computes the vector as an “average” vector from an individual content vectors),
and term frequency-inverse document frequency (TF-IDF) which is one of the most well-known measures for specifying keyword weights in Information Retrieval [16].

Let vectors $\vec{w}_u$ and $\vec{w}_s$ represent $UserProfile(u)$ of user $u$ and $Content(s)$ of document, the utility function $U(u,s)$ of the content-based approach can be represented by some scoring heuristic by using equation (2.17), such as cosine similarity [22].

$$U(u,s) = \cos(\vec{w}_u, \vec{w}_s) = \frac{\vec{w}_u \cdot \vec{w}_s}{\|\vec{w}_u\| \cdot \|\vec{w}_s\|} = \frac{\sum_{i=1}^{k} w_{iu} w_{is}}{\sqrt{\sum_{i=1}^{k} w_{iu}^2} \sqrt{\sum_{i=1}^{k} w_{is}^2}}$$  \hspace{1cm} (2.17)

where $k$ is the total number of keywords in the system.

Apart from traditional heuristics which are mainly based on information retrieval approaches, there are some other techniques that have been applied in content-based recommendations, such as modeling users by using machine learning, statistic learning [20], and probabilistic techniques [57]. For instance, in the bayesian classifier of probabilistic technique, the documents are rated as “relevant” or “irrelevant”. And each document is represented by a boolean vector that describes whether a certain term/keyword appeared in it. In the model, $P(k_i|C) = c$ expresses the probability of term appearing in a document labeled with keyword class [16]. The conditional probabilities are estimated by observing the training data. Its posterior probability $P(C = c |k_i)$ shows the probability that a document belongs to a certain class $c$ giving the keyword $k_i$. Given a set of keywords, $k_{1j}, k_{2j}, k_{3j}, k_{4j} ... k_{nj}$ in the document, the probability this document belongs to a class is expressed as $(C = c |k_{1j} \& k_{2j} \& ... \& k_{nj})$ [20].

In this research work, content-based approach is applied with a specific semantic reasoning technique in the process of matching of items descriptions for a better recommendation performance.

### 2.3.1.3 Hybrid Filtering Approach

Some systems started to use a hybrid approach to gain a better performance while reducing the limitations of any individual approach, such as [47, 58]. This trend had also been affected in Netflix Prize Competition [60], where different approaches were combined to improve the quality of recommendation results. The most common ingredient in hybrid approaches is CF combined with some other approaches, such as with the content-based approach.

Burke [43] investigates different hybrid recommendation approaches and proposed a taxonomy which classifies hybrid methods into seven classes. They are switching, weighted,
mixed, cascade, feature combination, meta-level and feature augmentation. The abbreviate definitions are shown in the Table 2.2 given below.

Table 2.2: Hybridization approaches - adapted from [43]

<table>
<thead>
<tr>
<th>Approaches for Hybridization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted</td>
<td>The scores (or votes) of several recommendation approaches are combined for producing a particular recommendation</td>
</tr>
<tr>
<td>Switching</td>
<td>The system switches among recommendation approaches based on the current situation</td>
</tr>
<tr>
<td>Mixed</td>
<td>Recommendations from numerous diverse recommenders are offered at one time</td>
</tr>
<tr>
<td>Feature combination</td>
<td>Features from diverse recommendation data sources are thrown together into a single recommendation approach</td>
</tr>
<tr>
<td>Cascade</td>
<td>One recommender refines the recommendations agreed by another</td>
</tr>
<tr>
<td>Feature augmentation</td>
<td>Output from one approach is used as an input feature to another</td>
</tr>
<tr>
<td>Meta-level</td>
<td>The model learned by one recommender is used as input to another</td>
</tr>
</tbody>
</table>

Jannach and Zanker [16] refer to the seven hybrid approaches mentioned above as hybridization strategies, and abstract them into three base designs from a more general perspective. They are monolithic, parallelized and pipelined hybrids. In parallelized hybridization design, the recommendation approaches involved can operate independently of each other and produce separate recommendation lists. The first three hybrid approaches in the table are referred to this type of design. A weighted hybrid recommender calculates the final score of an item by combing the results from all available recommendation approaches. A simple way can be linearly combining the scores from different recommendation approaches [43]. A switching hybrid method switches between recommendation approaches when certain condition is satisfied. For example, in a content/collaborative hybrid approach, if the content-based recommender cannot produce satisfied recommendation, a collaborative approach will be attempted [43]. A mixed hybrid recommender combines the results from different recommendation approaches together. Thus the recommendations from different approaches are combined as the final suggestions [43].

Monolithic hybridization design comprises a single recommender component that
integrates multiple techniques by pre-processing and combines different knowledge sources. It is
different from the other two hybridization designs which consist of two or more components
whose results are combined [16]. Two of the seven classes belong to monolithic hybridization
design. The feature combination uses the content-based approach but the data set it uses also
includes collaborative information, which is used as an added feature data connected with all
examples [43]. And the feature augmentation hybrid first applies one technique to acquire a
rating or a classification of an item, and then incorporates this data into further recommendation
process.

Pipelined hybridization design is when the output of a recommendation approach
becomes part of the input of another approach, such as cascade and meta-level hybrids [16].
Cascade hybrid recommender makes use of one approach to produce a coarse ranking of
candidates, and another approach for refining the initial candidate set [43]. In Meta-level hybrid,
the model generated by one approach is used as an input for another technique. The difference
between meta-level hybrid and feature augmentation is that the latter only takes the features or
information generated from an approach as the input for another one, while meta-level takes the
whole model as the input [43].

In this research, feature augmentation hybrid is applied for recommending WS to the
active user. First similar WS are clustered and then the proposed Scope One method is applied in
clusters for predicting QoS values. Other approaches are multi-criteria recommender systems
and context-aware recommender systems and out of scope to this research work.

2.3.2 General Limitations of Recommender System
Each type of recommendation technique has strengths and weaknesses, well known in the field.
The main characteristics of each technique are largely dependent on the source of information
being used. In this section, the limitations of each technique are analyzed. Furthermore, although
ideally HF techniques would overcome the problems of the combined techniques, there are
certain limitations that are inherent to the recommendation problem, and thus, have to be
addressed independently. Besides, by combining different methods, additional problems, along
with more limitations, arise.

The limitations identified in the literature for the main types of recommenders described
in the previous sections are given in the Table 2.3.
Table 2.3: Limitations of content-based and collaborative filtering approaches

<table>
<thead>
<tr>
<th>Limitations</th>
<th>Description</th>
<th>CBF</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restricted Content Analysis</td>
<td>Items to be recommended must have available data related to their features. This data is often unavailable or incomplete.</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Overspecialization</td>
<td>CBF recommenders are trained with the content features of the items. All the recommended items are similar to those already rated.</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Portfolio Effect</td>
<td>CBF recommenders suggest items based on the item features. An item is recommended even if it is too similar to a previously rated item.</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>New User</td>
<td>A user has to rate enough items in order to infer their preferences. When a new user enters into the system she has no ratings.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>New Item</td>
<td>Items have to be rated by a substantial number of users for being recommended. Recently incorporated items have none or insufficient ratings.</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Grey Sheep</td>
<td>A user has to be similar to others in the community to receive recommendations. Users whose tastes are unusual may not receive useful suggestions.</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Rating Data Sparsity</td>
<td>Ratings are used to train user and item models. The number of available ratings is usually small.</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

2.3.3 Evaluation Metrics of Collaborative Filtering

The quality of RS is decided based on the result of evaluation. Various types of metrics are used based on the type of CF applications. Herlocker et al. [52] classifies the metrics for evaluating recommendation systems as predictive accuracy metrics, such as Mean Absolute Error (MAE) and its variations; classification accuracy metrics, such as precision, recall, F1-measure, and ROC sensitivity; rank accuracy metrics, such as Pearson’s product-moment correlation, Kendall’s Tau, Mean Average Precision, half-life utility and normalized distance-based performance metric [58].

The commonly used CF metrics are MAE, Normalized Mean Absolute Error (NMAE) and Root Mean Squared Error (RMSE) and they are discussed below. For additional CF performance metrics of recommendation quality, see [16]. There are other evaluations of recommender systems including usability evaluation [59] and so forth.

**MAE and NMAE:** As an alternative to classification accuracy or classification error, the most frequently used metric in CF research literature is MAE [51, 52], which computes the average of
the absolute difference between the predictions and true ratings using the Eqn. 2.18.

$$MAE = \frac{\sum_{i,j} |p(r_{i,j}) - r_{i,j}|}{N}$$

(2.18)

where N is the total number of ratings over all users, \(p_{i,j}\) is the predicted rating for user \(i\) on item \(j\), and \(r_{i,j}\) is the actual rating. The lower the MAE value, the better the prediction.

Different recommender systems may use different numerical rating scales. NMAE normalizes MAE to express errors as percentages of full scale [51] and it is computed using Eqn. 2.19.

$$NMAE = \frac{MAE}{r_{\text{max}} - r_{\text{min}}}$$

(2.19)

where \(r_{\text{max}}\) and \(r_{\text{min}}\) are the upper and lower bounds of the ratings.

RMSE: RMSE which is becoming popular because it is the Netflix prize [60] metric for movie recommendation performance and it is computed using the Eqn. 2.20.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i,j} (p_{i,j} - r_{i,j})^2}$$

(2.20)

where n is the total number of ratings over all users, \(p_{i,j}\) is the predicted rating for user \(i\) on item \(j\), and \(r_{i,j}\) is the actual rating again. RMSE amplifies the contributions of the absolute errors between the predictions and the true values.

Although accuracy metrics have greatly helped the field of RS, the recommendations that are most accurate are sometimes not the ones that are most useful to users, for example, users might prefer to be recommended with items that are unfamiliar with them, rather than the old favorites they do not likely want again.

2.4 Chapter Summary

In this chapter, the basic definitions for Web service, their description, discovery, and invocation, along with their main standards are presented. As mentioned above, the standards on which WS are built because the main challenges surrounding the utilization of WS. These challenges are discussed according to each Web service standard:

- WSDL description depends on syntactic information only. It does not support semantics, nor does it support providing information related to the service’s QoS. In addition to that, it does not impose any constraints on the providers for supporting a textual documentation about the service functionality, nor for following naming conventions.
• UDDI discovery has failed to be the primary discovery mechanism. Alternatives are embodied in WS search engines and providers’ catalogs. They support searching by keywords only, which may not be adequate for efficient and precise service retrieval.

Especially that WSDL interfaces have a small quantity of text inside them, according to which, they can be indexed. These search engines do not allow searching for a needed operation, or searching along with QoS constraints. Thus, service selection considering current mechanisms and circumstances is neither efficient nor practical. In the following chapter, the literature’s proposed solutions for this problem is discussed, with their advantages and shortcomings.

RS start from mid-90s. They take users’ opinions or the interaction histories with the systems as the input, recommend the users the items they might prefer and offer the users with the information to help them to make a decision. The early most successful systems rely on the overall ratings representing the preferences to the items from user communities.

Pure content-based systems have known limitations, one of which is that any two items are indistinguishable if they are presented by the same set of keywords/features. Neither can it differentiate a good and a bad item, such as a well-written article and a very poor article. Additionally, it cannot predict some other items that the user happens to be interested but not known by himself/herself or shown before. Again, in order to cluster features of items easily, the content needs to be either in a form that can be automatically parsed by a computer (e.g., text) or can be assigned to items manually. Therefore, the content-based approach only clusters some of the characteristics. Some content information like visual content cannot be captured by content-based approach as by CF approach.

Another famous limitation is called overspecialization. The items recommended are those similar to the ones which have been positively rated by the user. This can lead to an undesirable effect such as obvious recommendations. In other words, the recommended items are too similar to those the user already knows. A typical example is a news recommender system that keeps recommending the same story to a user who has already read it before. As in these systems, the similarity metrics are based on syntactic approaches which only detect the resemblance between items sharing some or even all of the attributes.

There are a few ways of alleviating this overspecialization problem. Besides combining content-based systems with CF systems, some researchers introduce semantic reasoning
technique from semantic web to reduce the syntactic limitations of the traditional content-based approach. The reasoning process can infer the semantic associations between users’ preferences and items descriptions. Such associations allow a more flexible comparison between users’ profiles and items’ profiles as well.

CF suffers many limitations as well. Sparsity problem is a well-known one. The data are quite small comparing with what the system needs to predict. In particular, if the number of users is relatively small comparing to the large volume of items information (since there is always a very large or rapidly changing database, the user-item matrix will be very sparse. As a result, the number of users who share a sufficient number of rated items is small.

The cold-start problem is a significant problem in CF approaches, which includes new user and new item problem. Generally, CF approaches need a large amount of data to initiate it and to get a reliable result. The system has to learn the user’s tastes from his/her ratings history, which means it needs a sufficient number of items rated by this user before it works efficiently. Consequently, a new user may not be able to get accurate recommendations due to the lack of the rating history, which is the new user problem. Similarly, the recommendation quality of a new item would reach a reasonable level until it is being rated by a substantial number of users due to collaborative systems reply on users’ ratings to find their preferences.

In general, the content-based approach relies on rich information on item features and textual descriptions. The CF approach is more suitable in recommending when there are users sharing many rated items, which works best for a user who has many neighbors with similar tastes. However, if a system only applies the CF approach, it is not capable to recommend a newly added item, etc. The recommendations are also limited to the items that are alike to previously rated items, etc. As a result, researchers usually combine these two approaches together.

This research work attempts to propose a hybrid approach combining content-based and CF approaches to recommend services to the consumer which overcomes cold-start problem. Also proposes a new method to forecast the QoS values with better forecasting accuracy.

The literature survey related to the proposed research work is discussed in the next chapter.