Project Number: **215219**

Project Acronym: **SOA4ALL**

Project Title: **Service Oriented Architectures for All**

Instrument: **Integrated Project**

Thematic Priority: **Information and Communication Technologies**

D2.7.2 - Recommender System Second Prototype

Activity:	Activity 1 - Fundamental & Integration activities	
Work Package:	WP2 - Service Deployment and Use	
Due Date:	31/08/2010	
Submission Date:	31/08/2010	
Start Date of Project:	01/03/2008	
Duration of Project:	36 Months	
Organisation Responsible of Deliverable:	CEFRIEL	
Revision:	1.0	
Author(s):	Daniele Dell'Aglio	CEFRIEL
	Irene Celino	CEFRIEL
	Dario Cerizza	CEFRIEL
	Liwei Liu	UNIMAN
	Freddy Lecue	UNIMAN
Reviewer(s):	Sven Abels	TIE
	Nikolay Mehandjiev	UNIMAN

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	13/07/2010	Document Initialized	Irene Celino
0.2	20/07/2010	Revised table of contents	Irene Celino
0.3	30/07/2010	Executive summary, introduction, chapter 1 and chapter 2	Irene Celino and Daniele Dell'Aglio
0.4	02/08/2010	Contribution to chapter 3	Irene Celino and Daniele Dell'Aglio
0.5	03/08/2010	Contribution to chapter 2, references and draft of conclusions	Irene Celino and Daniele Dell'Aglio
0.6	04/08/2010	Contribution to Chapter 3	Liwei Liu and Freddy Lecue
0.7	04/08/2010	Contribution to Chapter 4	Daniele Dell'Aglio
0.8	05/08/2010	Contribution to Chapter 4	Freddy Lecue and Liwei Liu
0.9	06/08/2010	Finalization of Chapter 4 and check of the whole document	Daniele Dell'Aglio and Irene Celino
0.91	23/08/2010	Integration of comments from reviewer: Sven Abels (TIE)	Irene Celino
0.92	23/08/2010	Integration of comments from reviewer: Nikolay Mehandjiev (UNIMAN)	Irene Celino, Liwei Liu
1.0	31/08/2010	Finalization and delivery of the document to the EC	

Table of Contents

VERSION HISTORY 2
TABLE OF CONTENTS 3
EXECUTIVE SUMMARY 7
1. INTRODUCTION 8
1.1 PURPOSE AND SCOPE 8
1.2 STRUCTURE OF THE DOCUMENT 8
2. OVERVIEW OF THE RECOMMENDER SYSTEM IN SOA4ALL 9
2.1 RELATION WITH THE SOA4ALL ARCHITECTURE 9
2.2 RS API 10
3. INSIDE THE RECOMMENDER SYSTEM 12
3.1 ADOPTED APPROACH 12
3.2 THE COLLABORATIVE FILTERING RS 13
3.3 THE SEMANTIC WEB ENABLED KNOWLEDGE-BASED RS 14
3.3.1 Linked Data-driven Recommendations 14
3.3.2 Our Semantic Web-enabled Recommender system 16
3.4 THE SEMANTIC CONTENT-BASED RS WITH CONTEXT CONSIDERATION 17
3.4.1 Background 17
3.4.2 Recommendation Generation 19
4. ARCHITECTURE, INSTALLATION AND CONFIGURATION 23
4.1 ARCHITECTURE OF THE WHOLE RECOMMENDER SYSTEM AND INSTALLATION OF THE RS COMPONENT 23
4.2 INSTALLATION AND CONFIGURATION OF THE RS COMPONENT 24
4.3 INSTALLATION AND CONFIGURATION OF THE COLLABORATIVE FILTERING RECOMMENDER SYSTEM 24
4.4 INSTALLATION AND CONFIGURATION OF THE SEMANTIC WEB ENABLED KNOWLEDGE-BASED RECOMMENDER SYSTEM 24
4.5 INSTALLATION AND CONFIGURATION OF THE SEMANTIC CONTENT-BASED RECOMMENDER SYSTEM 25
5. CONCLUSIONS 26
REFERENCES 27
ANNEX A. LIST OF PAPERS 29

List of Figures

Figure 1 – Relation between the RS and the Consumption Platform 9
Figure 2 – Components of the recommender system 12
Figure 3 – Architecture of Collaborative Filtering and Semantic Web-enabled RS 14
Figure 4 – General architecture our Semantic Web-enabled Recommender System 15
Figure 5 - Part of an *ALE* TBox 18
Figure 6 - Context Taxonomy 21
Figure 7 – Recommender System Architecture and its integration in SOA4All 23
Figure 8 – Semantic Web enabled Recommender System architecture 24

List of Tables

Table 2.1 – Batch-time RS API methods.....10
Table 2.2 – Run-time RS API methods. 10
Table 2.3 – The RecommendedService class returned by the RS API.....11

Glossary of Acronyms

Acronym	Definition
AP	Analysis Platform
API	Application Programming Interface
CP	Consumption Platform
DL	Description Logic
FOAF	Friend Of A Friend
GUI	Graphical User Interface
LarKC	Large Knowledge Collider
LOD	Linking Open Dataset
OWL	Web Ontology Language
OWL-S	Web Ontology Language for Services
QoS	Quality of Service
RDF	Resource Description Framework
REST	Representational State Transfer
RIF	Rule Interchange Format
RS	Recommender System
RSC	Recommender System Component
SA-WSDL	Semantic Annotations for WSDL
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SPICES	Semantic Platform for the Interaction and Consumption of Enriched Services
SWRL	Semantic Web Rule Language
SWS	Semantic Web Service
TF-IDF	Term Frequency–Inverse Document Frequency

Executive summary

This document complements the release of the second Recommender System (RS) prototype by accompanying the software prototype of the Recommender System, as the result of the activities performed in the scope of T2.7. The software release contains the source code, the installation and configuration facilities.

The second version of RS aims to support SOA4All users by providing suggestions (a.k.a. recommendations) about services that they may be interested in. The RS is based on a set of different algorithms and techniques that exploit the available information about services, users and their behaviour within SOA4All. Details on those techniques are given in this document.

The body of this document follows the project guidelines for prototype releases, thus it reports the description of the component and the installation and configuration activities. This deliverable provides also an overview of the approach adopted, the architecture defined and the integration performed to provide a recommender system in SOA4All. More details on the scientific contributions are also provided in the form of papers submitted/accepted to relevant conferences.

The structure of the document follows the table of contents of the previous deliverable D2.7.1, but each chapter has been revised to add the up-to-date information. In particular, Chapter 2 refines the role of the RS in the general SOA4All architecture and details the updated API; Chapter 3 refines the overall approach by adding the new algorithms and techniques added in the second prototype; Chapter 4 integrates the previous installation instructions by explaining the differences and the novelties introduced in the second prototype.

1. Introduction

1.1 Purpose and Scope

This deliverable illustrates the Recommender System (RS) integrated within the SOA4All Consumption Platform. This system aims to improve the user experience by providing users with suggestions about relevant services that may be of their interest. In order to provide recommendations, the RS analyzes different kinds of data: service descriptions, user profiles and user behaviour in interacting with the platform.

M30 release constitutes the final release of the RS component and extends the first version released at M18 by adding new algorithms to compute recommendations. In particular, we explored the possibilities to leverage the semantic descriptions of services and users in order to improve the recommendations.

The goal of this deliverable is to complement the Recommender System software prototype.

1.2 Structure of the document

In Chapter 2 we position the RS within the SOA4All architecture (Section 2.1) and we detail the RS API through which the SOA4All Consumption Platform can invoke the RS functionalities (Section 2.2).

Chapter 3 is devoted to explain how the RS works. In Section 3.1 we illustrate the general approach and why we decided to explore more than one algorithm/technique to compute recommendations; the following Sections 3.2-3.4 contain a description of each of the employed approaches, in order to let the reader understand the different ways we compute recommendations.

Chapter 4 presents the instructions to install and configure the RS within the SOA4All Consumption Platform; this is because the developed code is released under an open source licence to let people outside the SOA4All consortium experiment with our scientific and technical results.

Finally, Chapter 5 reports some conclusions and Annex A lists a set of scientific papers accepted or under review at different conferences that prove the value of our work.

2. Overview of the Recommender System in SOA4All

The objective of this chapter is to illustrate the role of the Recommender System in SOA4All. In particular, we describe the relation between the Recommender System and the other main components of the Consumption Platform.

2.1 Relation with the SOA4All Architecture

Interacting with services in a service world as the one envisaged by SOA4All requires implementing several mechanisms in order to enhance the user experience within the vast number of services expected. In particular, enabling ways to help end-users to interact with the most suitable services for them is a challenge, for while it is obviously an advantage to have many services to choose from, there is a need to enable methods to find the most appropriate ones. Recommendations will be one of these mechanisms that will permit users to be aware of items (i.e. services, specifically for SOA4All) that can be helpful for them.

The Recommender System is the key component responsible for providing useful recommendations for users. These recommendations actually take place in the Consumption Platform, thus there is a strong relation of the RS component with that platform, which will query the RS for relevant recommendations.

It is worth noting that the recommendations will be useful not only for helping users to find relevant services by itself, but also because these recommendations take place in an active mode, and this is expected to improve the user experience within the platform, hence making them more bound to engage within SOA4All.

It is also important to point out that the relation of the RS with the Consumption Platform is not only mono-directional (the outcomes of the RS benefiting the platform), but it is bi-directional: the RS needs to know the interactions of users within the platform and the basic description of services and users available in the platform and in other SOA4All components.

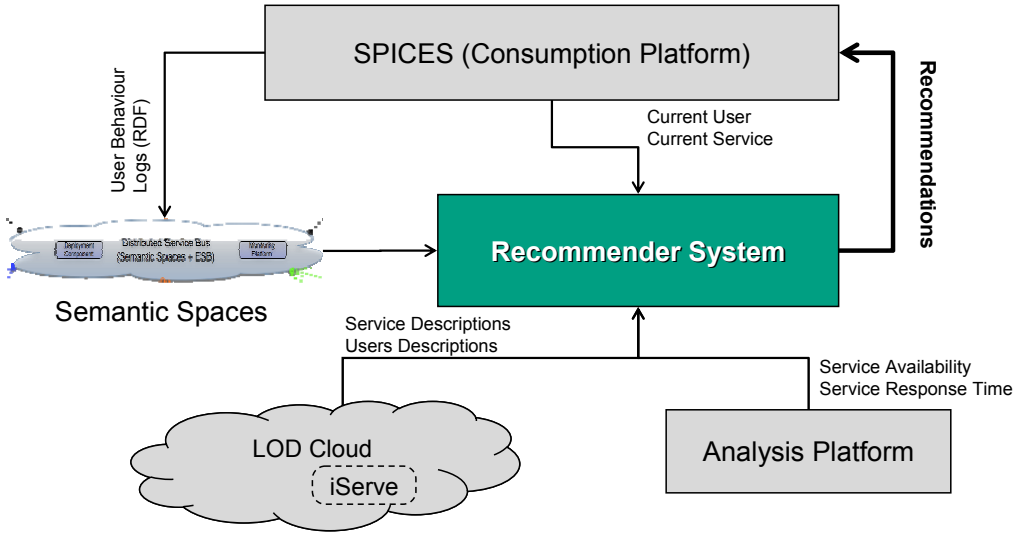

Figure 1 – Relation between the RS and the Consumption Platform

Figure 1 depicts the relation between the Recommender System and the other SOA4All components.

The RS receives as inputs a number of information about services (their semantic description from iServe, the availability/response time statistics from the Analysis Platform) and about users (some profile data from the Consumption Platform itself, the logs of users' interactions with SOA4All from the Semantic Spaces and some additional data from the "Linking Open

Data dataset cloud" or simply "LOD Cloud"[1]). Based on those inputs, the RS is able to compute recommendations, which are stored in some internal dedicated data structures; whenever some more information is added (e.g. additional user logs), the RS processes the new input and updates the recommendations.

The RS provides recommendations to the Consumption Platform whenever requested; SPICES passes the user id and/or the service id to the RS, which accordingly returns a list of suggested services together with a score representing the "confidence" of each recommendation and, in the case of Semantic Web-based suggestions, an explanation of the recommended service.

2.2 RS API

The RS exposes its functionalities through a well-defined API, which hides the complexity and the different algorithms adoption inside the component. The API is constituted by a batch-time part and by a run-time part.

The *batch-time API* refers to the operations of the RS performed prior to the actual proposition of service recommendations to the user. The batch-time interface is included in the `it.cefriel.swa.rs.api.RecommenderSystemBatchTime` class; its main methods are listed in Table 2.1.

```
public abstract void start();

public abstract void reset();

public abstract Date getLastLogEntryDateAdded();

public abstract void addLogEntrySet(URI logEntrySetUri, Date creationDate,
            String creator, Set<LogEntry> logEntrySet);
```

Table 2.1 – Batch-time RS API methods.

Apart from the `start()` and `reset()` methods, which respectively starts the batch time of the RS and resets all the tables, the `addLogEntrySet()` method allows to insert a log entry set into the RS to let it compute new recommendations, while the `getLastLogEntryDateAdded()` method returns the date on which the last log entry set was inserted (in order to retrieve from the Semantic Spaces only the new log entries, when re-computing the recommendations).

The run-time API refers to the functionalities offered by the RS to retrieve service recommendations; the Consumption Platform invokes the RS to get and then display the recommendations to the user. The run-time methods, listed in Table 2.2, are included in the `it.cefriel.swa.rs.api.RecommenderSystemRunTime` class:

```
List<RecommendedService> getRecommendationByService(URI service, int num);

List<RecommendedService> getRecommendationByUser(URI user, int num);

List<RecommendedService> getRecommendationByUserAndService(URI user, URI
            service, int num);
```

Table 2.2 – Run-time RS API methods.

[1] Cf. http://richard.cyganiak.de/2007/10/lod/.

The three methods above hide the different algorithms (described in the following Chapter 3) behind the same interface. There are three different ways the Consumption Platform can ask the RS for recommendations:

- when a user is not logged in but he/she is analyzing a specific service, the RS can provide recommendations only on the basis of the description of the current service; in this case the `getRecommendationByService()` method is invoked;
- when a user is logged in but he/she is not analyzing any specific service, the RS can provide recommendations only on the basis of the profile of the current user; in this case the `getRecommendationByUser()` method is invoked;
- when a user is logged in and he/she is analyzing a specific service, the RS can provide recommendations on the basis of both the description of the current service and the profile of the current user; in this case the `getRecommendationByUserAndService()` method is invoked.

Finally, to complete the overview of the RS API, we describe the characteristics of the recommendations returned by the RS run-time under the form of objects belonging to the `it.cefriel.swa.rs.api.RecommendedService` class in Table 2.3.

```
public class RecommendedService extends WeightedObject<URI, Float>
{
   // fields inherited from WeightedObject
   private URI object;
   private Float_strength;

   // own fields
   private String proof;
}
```

Table 2.3 – The RecommendedService class returned by the RS API.

A `RecommendedService` object points to the URI of the service to be recommended, gives a float number which represents the score, the “utility value” of the recommendation and, in case of knowledge-based recommendations, provides also a proof, i.e. a textual explanation of the reason why that service is suggested to the user.

3. Inside the Recommender System

The Recommender System (RS) in SOA4All aims to provide users with recommendations about services (the recommended items) that could be of their interested. This functionality acts as an additional feature to support users in discovering services that meet their needs.

This chapter describes the RS in SOA4All from a functional perspective, starting with the general approach adopted, the various algorithm and techniques adopted and the different recommender implemented and integrated within the RS component.

3.1 Adopted approach

Recommender systems are becoming more and more commonly used to help users serendipitously find items they were (implicitly or explicitly) looking for. From a user's point of view, recommendations are seen as suggestions that are proactively provided by the system, in a timely fashion. In order to be effectively useful, the recommendations should be accurate, as to "foresee" a user's needs.

Recommender systems are usually classified by the recommendation technique they use [16]:

- *Collaborative Filtering Recommender Systems* [17]: given a user, they find users with similar behavior to predict items of interest;
- *Knowledge-based Recommender Systems* [5]: they build a knowledge base with a model of the users and/or items in order to apply inference techniques and find matches between users' need and items' features;
- *Content-based Recommender Systems* [18]: they usually employ a classifier to predict items' similarity.

Additionally, another category of systems, the *Hybrid Recommender Systems* [16], tries to join the advantages of two or more techniques described above.

The approach we followed in the SOA4All project consists in building a Hybrid Recommender System (see Figure 2), composed by a number of different recommenders, which explored the various possibilities enabled by the distinct algorithms and techniques.

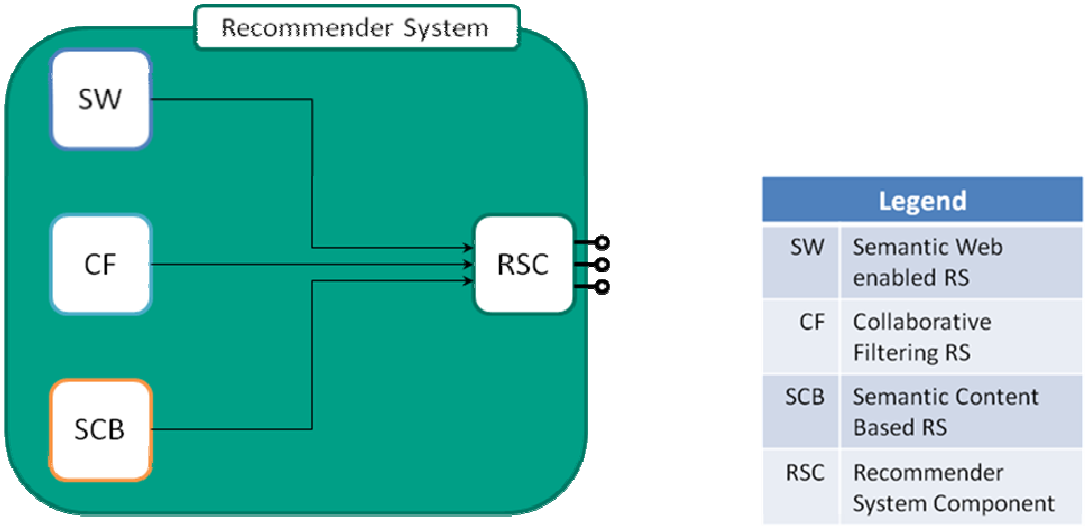

Figure 2 – Components of the recommender system

In Sections 3.2-3.3-3.4, we therefore illustrate the individual recommenders we built on top of

the available information provided by SOA4All (service descriptions, user profiles, execution logs, etc.): respectively the Collaborative Filtering RS, the Semantic Web enabled and the Semantic Content Based RS. For each recommender, we explain the algorithm and the employed information and we illustrates the benefits brought to the Recommender System in the whole.

In the following of this section, we briefly explain the approach followed to integrate them into a single Hybrid Recommender System.

As detailed in Section 2.2, the RS component as a whole exposes a well-defined API to the rest of SOA4All components. Such an API was designed to provide the recommendation functionalities transparently with respect to the specific algorithm adopted. This is the reason why the first prototype of the RS component, illustrated in D2.7.1 [1], included a collaborative filtering approach, while the second prototype combines three distinct techniques, providing however the same consistent API.

As such, the Hybrid Recommender System built on top of those distinct recommenders "hides" its complexity to the other components. In this way, we were able to combine the strength points of each recommender in order to improve the overall behaviour of the RS component.

For example, while the Collaborative Filtering Recommender (detailed in D2.7.1 [1] and summarized in Section 3.2) is able to implement all the three methods listed in Table 2.2, the Semantic Web-enabled Recommender (illustrated in Section 3.3) leverages on the user profiles and therefore can provide a valuable contribution to the user-based methods. The approach presented in Section 3.4 is able to implement the first method of Table 2.2, namely `getRecommendationByService`, focusing then on context and service description to recommend services. Similarly, the recommendation "proof" (as described in Table 2.3) is provided only by the Semantic Web-enabled Recommender, since it is the only technique able to grant such explanation of its suggestions.

3.2 The Collaborative Filtering RS

The first algorithm that we employed in the RS component follows a Collaborative Filtering approach. The details about this technique are fully described in deliverable D2.7.1 [1] and in the paper [2], to which the reader should refer for further information.

With regards to Figure 3, the collaborative filtering algorithm leverages the User Behaviour Analyzer which processes the user logs stored in the Semantic Spaces every time a SOA4All user interacts with the platform.

A thorough evaluation of the Collaborative Filtering Recommender System component was also performed and the results are illustrated in [3] and [4].

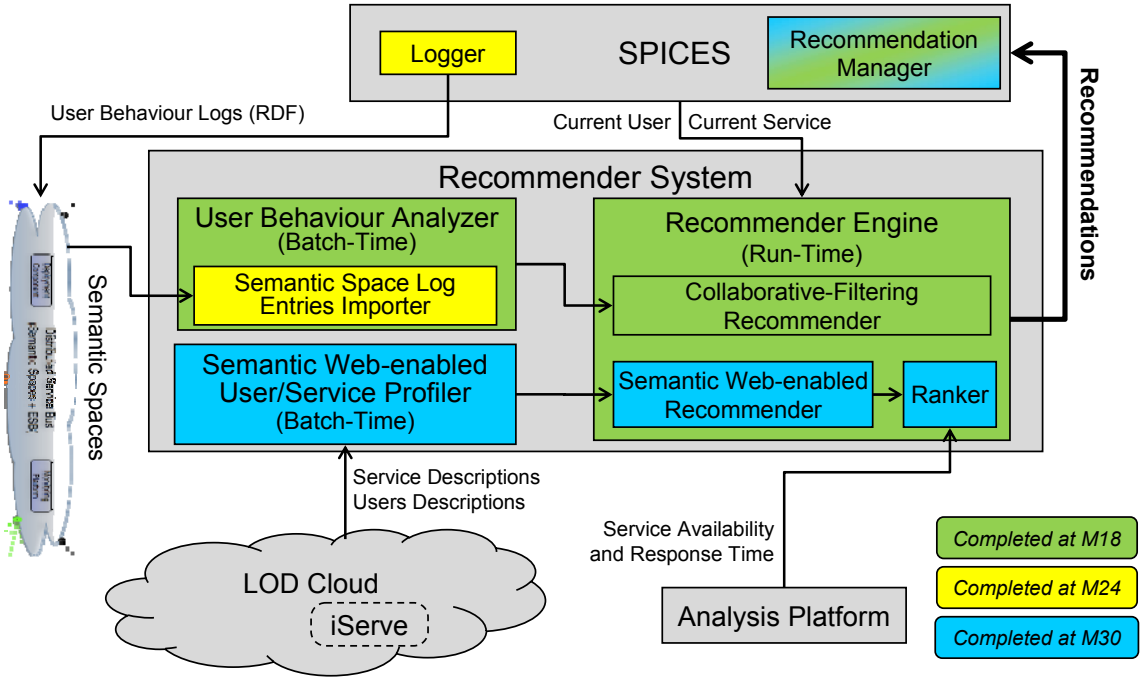

Figure 3 – Architecture of Collaborative Filtering and Semantic Web-enabled RS

3.3 The Semantic Web enabled Knowledge-based RS

As introduced above, within the second version of the Recommender System component includes also a different recommender based on a knowledge-oriented approach that employs Semantic Web technologies.

With regards to Figure 3, this second recommender derives semantic descriptions of users and service by interacting with a number of components, both internal to SOA4All (the Consumption Platform, iServe, the Analysis Platform) and external to it (like the open linked data Web, namely the LOD Cloud).

In the following, we explain the basic ideas and we give some details about the implementation of such a component.

3.3.1 Linked Data-driven Recommendations

Knowledge-based recommender systems [5] are systems that select items of interest for users, by analyzing items' features (and optionally users' profiles) stored in a knowledge-base. The advantages of this kind of recommender systems are mainly: (1) *minimal amount of users*: unlike collaborative recommender systems, this kind of systems does not require a huge amount of users to compute recommendations; (2) *no cold start*: when a new user/item is added with its description, the system does not suffer of the cold start problem: it is immediately able to compute recommendations for the new user/item; (3) *proof-generation for the recommendations*: it is possible to explain the motivation behind an item proposal.

One of the main problems of this category of recommender systems is that the knowledge base has to be created and maintained. The creation requires several steps: the domain should be modelled; each user and item should be described according to the model; a set of policies should be defined to compute recommendations. The knowledge base also requires to be maintained over time: the domain could change, requiring the modification of the model; the recommendation policies could be revised and so on. Usually these operations require a lot of effort, with a heavy human intervention.

We believe that the Web of Data should be considered as an interesting source of information to be used by Knowledge-based Recommender Systems. The LOD Cloud is a huge public source where information can be found to describe several kinds of items, users and domains. Accessing the Web of Data and exploiting Semantic Web technologies can allow for the partial automation of the *knowledge base creation and maintenance*, simplifying the modelling and profiling of items and users. Furthermore, the computation tasks to generate recommendations, operating on the knowledge base, can be performed and enhanced by the use of Semantic Web tools, such as reasoners or rule-based systems, exploiting the potentialities of standard languages like SPARQL, RIF and so on.

In Figure 4, we represent our concept of Semantic Web-enabled Knowledge-based Recommender System.

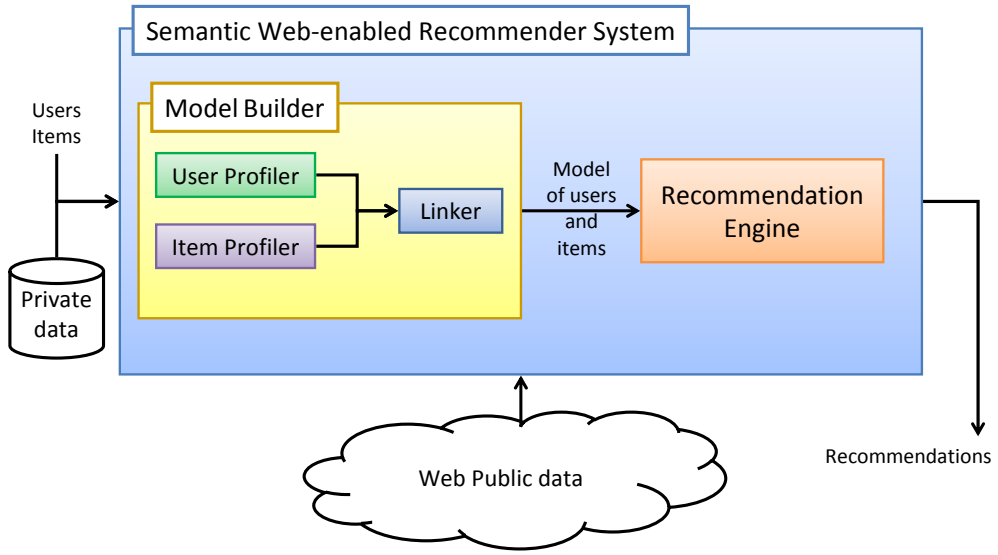

Figure 4 – General architecture our Semantic Web-enabled Recommender System

The *Model Builder* is the sub-system devoted to build the Recommender System knowledge base by reconstructing a description of items and users; it takes information both from the application "private data" (e.g. items list with some characteristics, users identifiers) and the public Web, in particular the Web of Data.

Within the Model Builder, an *Item Profiler* component is included to get additional information to describe the items of interest; for example, it tries to retrieve classifications or categorizations from the Web of Data that can enrich the available private data. In order to do this, it queries Semantic Web search engines (like Sindice [6] or Watson [7]) and look for additional data from the LOD Cloud, for example from DBpedia [8].

Similarly, a *User Profiler* component enriches the users profiles by retrieving additional information about their interests and preferences, both from the Social Web (e.g. from social networks) and from the LOD Cloud, in particular by getting FOAF descriptions [9].

Finally, after the reconstruction of users and items profiles, a *Linker* component finds information about how users (and their interests) are related to items (and their features). To do so, it queries the LOD Cloud by looking for "semantic paths" that connect the data included in the user profiles to the data describing the items, i.e. it tries to put in relation the outputs of the two Profilers; this is possible because the Web of Data – like the Web of Documents – has a graph structure that can be traversed to find connections.

During the course of the SOA4all prototype, the query connectors to the following Semantic Web SPARQL endpoints or search engines have been setup: Sindice[2], DBpedia[3], OpenLink endpoint[4] and FactForge endpoint[5].

But Semantic Web technologies can be employed also to compute recommendations. This is why also the final component in Figure 4 – the *Recommendation Engine* – leverages the potentialities of the Semantic Web. Operating on the knowledge base created by the Model Builder, it computes all the possible paths connecting a user to an item, and it "evaluates" those paths to give them a utility value; the assumption is that, if a user profile can be connected to an item description by a set of semantic links, this means that that item is a good candidate for recommendation. The Recommendation Engine evaluation is aimed to assess the "*meaningfulness*" of such connections; by the employment of Semantic Web tools like SPARQL processors [10] or rule systems using SWRL [11] or RIF [12], it verifies a set of constraints within the computed path; e.g. it looks for semantic links expressing interest, liking or importance (e.g. the user likes a topic which is related to the item) or, on the contrary, for expressions of disapproval or distaste (e.g. the user dislikes a subject to which the item refers).

3.3.2 Our Semantic Web-enabled Recommender system

On the basis of the general architecture of the Semantic Web-enabled Recommender System presented in the previous section, we developed a prototype of the above concept to recommend Web services (the items of interest) to SOA4All developers (final users).

The application was developed over the LarKC platform [13][14]: this means that we designed our recommender system as a LarKC "workflow", i.e. a set of software plug-ins with Semantic Web capabilities, executed in a certain order and passing data among them. This let us reuse existing plug-ins and realize modular code.

Regarding the users, our prototype receives as input a user identifier and then retrieves his interests from the Web. Since in the SOA4All environment a user is identified by his OpenID, our system looks for sources pointing to such identifier, e.g. an available FOAF profile [9]. When such a profile is retrieved, our system tries to extract further information from the Web, in order to collect useful hints to identify users' preferences. It is worth noting that, since the Web sources can employ different schemata to describe the user, we employed ontology mappings to re-conduct the collected data into a common format, using FOAF and the Weighted Interest ontology[6].

A similar approach was followed to profile the services, in order to retrieve useful information for the recommendation computation (service categorization, information about QoS, etc.) and to build a uniform description of services. Starting from the semantic description of services retrieved from iServe [15], the service profiler analyzes its categorizations and tries to find further links to linked data resources, e.g. to DBpedia categories and topics.

Finally, elaborating the collected knowledge, our system looks for paths between users and services and the Recommendation Engine component evaluates them by attributing a "utility value" based on their content. The found path will also serve as "proof"/explanation of the

[2] Cf. http://sindice.com/.

[3] Cf. http://dbpedia.org/.

[4] Cf. http://lod.openlinksw.com/.

[5] Cf. http://ldsr.ontotext.com/.

[6] Cf. http://xmlns.notu.be/wi/.

recommendation: when the suggestion is displayed to the user, he can visualize such an explanation to understand why he got the specific service suggestion. This in turn enables a direct evaluation of the Recommender System results, since the user can explicitly say if he likes/dislikes the recommendation and he can complement/improve his profile or the service description, in order to let the system produce more effective and meaningful suggestions.

The interest reader can refer to the papers cited in Annex A for further details.

3.4 The Semantic Content-based RS with Context Consideration

Currently, web service marketplaces and search portals such as XMLMethods, BindingPoint, and WebServiceList, are immature and do not provide the wealth of user feedback, reviews and rankings which characterize their mature counterparts focused on products (i.e. PriceRunner, Amazon) or even conventional services (i.e. TripAdvisor, epinions). This lack of user feedback is also known as the "cold-start problem". Our approach aims at proposing a content-based recommendation technique using semantic similarity measures to solve this problem. In more details, we use semantic content based approach based on semantic similarity and context based information.

3.4.1 Background

3.4.1.1 Content-based Approach

This approach is based on one of the classical approach in recommender system area, which is content-based approach. Generally, content-based approach recommends those items which are similar to the ones the user preferred in the past. To do this, it describes the items that may be recommended, and creates a profile of the user that describes the types of items the user likes, and then compares items to the user profile to determine what to recommend [18]. The item can be described through the same set of attributes, or by some attributes with a set of restricted values and some free-text fields when the domain is semi-structured, or in a text-based area, by using TF-IDF etc techniques [18]. On the other hand, the user can be modelled either through the description of themselves, or by collecting user's view history.

Here, the "item description" will be semantic web service description, while user model is built based on the user's view history.

3.4.1.2 Semantic Web Service Description

As semantic specifications of Web services are used in our content-based RS. Here we review i) service descriptions, and ii) a non standard DL (Description Logic) reasoning techniques we used to infer the commonality and differences in service descriptions.

3.4.1.2.1 Semantic Web Services Descriptions.

The formal model required to represent semantics of a web service *s* is defined as a set of semantic attributes:

– its *functional category* $F(s)$;

– its *functional parameters* i.e., inputs $In(s)$ and outputs $Out(s)$;

– its requirements i.e., preconditions $P(s)$ and effects $\mathrm{E}(s)$;

All are provided by a domain ontology T through semantic annotations. The particular ontology T is based on the DL ALE [19], mainly defined by T its Terminological Box (or TBox i.e., intentional knowledge) in DL systems. In the following, the TBox T i) is used to annotate service descriptions, and ii) supports inference on these descriptions by means of

DL reasoning. Fig. 3 shows a fragment of an example TBox T .

According to this model, semantic web services require input parameters to be processed and preconditions to be satisfied and return some output parameters with some effects. In addition a (meta) semantic description related to its functional category is attached to each service, enabling to reason on its functionality and disambiguating services with similar functional parameters. From a semantic web service implementation view, the Minimal Service Model (http://cms-wg.sti2.org/TR/d12/v0.1/) is used to describe them.

$$
\begin{aligned}
&ProductDescription \equiv \forall hasDescription.ProductsList \sqcap \exists hasDescription.ProductsList\\
&FootwearProductDescription \equiv ProductDescription \sqcap \forall hasDescription.FootwearProductsList\\
&\qquad \sqcap \exists hasDescription.FootwearProductsList\\
&Product \equiv \forall hasID.ProductID \sqcap \exists hasID.ProductID \sqcap \forall hasName.ProductName\\
&\qquad \sqcap \exists hasName.ProductName \sqcap \forall hasURI.ProductURI\\
&\qquad \sqcap \exists hasURI.ProductURI\\
&ValidProductID \equiv ProductID \forall hasValid.ID \sqcap \exists hasValid.ID\\
&ProductURI \equiv \forall hasURI.ProductURI \sqcap \exists hasURI.ProductURI\\
&ProductData \equiv \forall hasID.ProductID \sqcap \exists hasID.ProductID \sqcap \forall hasName.ProductName\\
&\qquad \sqcap \exists hasName.ProductName \sqcap \forall hasURI.ProductURI\\
&\qquad \sqcap \exists hasURI.ProductURI \sqcap \forall hasPrice.ProductPrice\\
&\qquad \sqcap \exists hasPrice.ProductPrice\\
&ProductsList \sqsubset \top, ProductID \sqsubset \top, ProductPrice \sqsubset \top, Europe \sqsubset \top, ID \sqsubset \top, Format \sqsubset \top
\end{aligned}
$$

Figure 5 - Part of an ALE TBox

3.4.1.2.2 Common and Missing Description

Given the definition of semantic web service, RSs may suggest services, which have been consumed by similar end-users, based on their semantic similarity e.g., in terms of their functional parameters, categories and requirements. In this direction, the semantic similarities between two semantic descriptions sd_i, sd_j (referring to any attribute of service descriptions), encoded using the same TBox T , can be judged using a matchmaking function. This function enables finding some (basic) levels of semantic compatibilities [30][25][29] (i.e., *Exact*, *PlugIn*, *Subsume*, *Intersection*) and incompatibilities (i.e., *Disjoint*) among services, based on subsumption relationships.

Computing such basic semantic similarities can be completed with more detailed information i.e., the DL concept descriptions: *Missing* and *Common Descriptions* (first defined as the *Extra* and *Common Descriptions* in [24].

On the one hand the computation of *Missing Descriptions* is done by exploiting a non-standard DL reasoning: the *difference* or subtraction operation [20] for comparing ALE DL-based descriptions, thus obtaining a compact representation of the metric:

(i) the Missing Description $sd_j \setminus sd_i$

$$sd_j \setminus sd_i \equiv \min_{\circ_d}\{E \mid E \sqcap sd_i \equiv sd_j \sqcap sd_i\}. \qquad (1)$$

which refers, with respect to the subdescription ordering $\circ_d$ [23], to information required by sd_i to be semantically closer to sd_j . This defines all information which is a part of the description sd_j but not a part of the description sd_i. In case $T \models sd_i \sqsubseteq sd_j$, (1) refers to information which is required by sd_i to be similar sd_j. The *Missing Description* (1) is not only necessary to explain how two descriptions are different, but also why they are different and

how to make them (semantically) closer and even similar.

On the other hand, the *Common Description* of sd_i and sd_j is defined as:

(ii) their Least Common Subsumer [22] *lcs* as a DL concept description i.e.,

$$lcs(sd_i, sd_j) \equiv \{F \mid sd_i \sqsubseteq F \wedge sd_j \sqsubseteq F \forall F' : sd_i \sqsubseteq F' \wedge sd_j \sqsubseteq F' \Rightarrow F \sqsubseteq F'\}. \quad (2)$$

which refers to information shared by sd_i and sd_j.

3.4.1.3 Context Factor

Context pays an important role in making decisions, and the recommendation covering context is more precise and more personalized than recommendation without context. Researchers hold different opinions about the definition of context. In this part, we are introducing the definition provided by Mostefaoui and Hirsbrunner [28]. They propose a formal definition of web service context based on Dey's context definition. Context is 'any information that can be used to define the situation of an entity in a service-oriented environment'. And the entity here means 'a person, a sensor, a computing device, a service or any other object that can be considered relevant to the interaction between a user and a service'.

Maamar et al [27] classify context into three types, user context, web service context and resource context. User context is about user's location, previous activities and preferences. Web service context is about locations of execution, times of execution, and constraints during execution, by aggregating of its simultaneous participants in composite services. Resource context is referred as resource's current status, periods of non-availability, and capacities of meeting the execution requirements of web services. These three types related to each other [27].

In this work, we focus on web service context, since user's selection is decided by web services functions, which connect to the execution environment tightly. Although different web services can have different contexts, there are still some general ones which can cover the overall contexts. Here, we mainly focus on three contexts. They are: *intended use* (directly use; for composition), *use frequency* (high; median; low) and *type of use* (leisure; business/work; others). These three can affect the user's selection. For example, *usage frequency*, if it is used by a company, the handling data is potentially much larger than it used by an individual. Thus, the requirements on execution time and capacity are higher. *Intended use* is another context dimension. Directly use and used for composition may have different levels of requirements on availability.

3.4.2 Recommendation Generation

We extend Content-based approach to semantic content-based by introducing semantic similarity of web services. And context information is also modeled within our approach through measuring the similarity of contexts.

3.4.2.1 Semantic Similarity

One generic measure is considered for evaluating semantic similarity between services descriptions: their Common Description rate.

Definition 1 (Common Description rate):

Given two *ALE* semantic description sd_i and sd_j, the Common Description rate $q_{cd} \in (0,1]$ provides one possible measure for the degree of similarity between sd_i and sd_j. This rate is

computed using:

$$q_{cd}(sd_i, sd_j) = \frac{\left|lcs(sd_i, sd_j)\right|}{\left|sd_j \setminus sd_i\right| + \left|lcs(sd_i, sd_j)\right|}. \quad (3)$$

This rate estimates the proportion of description in sd_i and sd_j which are in common. The higher the better is the similarity. The expressions in between | refer to the size of *ALE* concept descriptions ([23] p.17) i.e., $|T|, |\bot|, |A|, |\neg A|$ and $|\exists r|$ is 1; $|C \sqcap D| = |C| + |D|$; $|\forall r.C|$ and $|\exists r.C|$ is $1 + |C|$. For instance |*ProductData*| is 22 with respect to Fig.3. The common description rate is pre-computed and provided through DL reasoning by [24].

Given the above quality criteria, the semantic similarity of two semantic descriptions sd_i and sd_j can be defined by equation (3) where sd_i and sd_j can be respectively any semantic attribute of service descriptions i.e., $In(s_i)$ and $In(s_j)$; $Out(s_i)$ and $Out(s_j)$; $\mathrm{E}(s_1)$ and $\mathrm{E}(s_2)$; $P(s_i)$ and $P(s_j)$; $F(s_i)$ and $F(s_j)$ of services s_1 and s_2. By considering this quality model, we aim at evaluating the level of semantic similarity between two different services descriptions.

In case some semantic attributes of services are defined by multiple semantic descriptions, the value of each quality criterion is retrieved by computing their average. In more complex cases, where the number of semantic descriptions are different between attributes of services, only comparable (in term of subsumption) pairwise of descriptions are considered.

The quality model (3) for semantic similarity can be generalized to any pair of services s_1 and s_2 rather than to any pair of semantic descriptions (or services attributes) as following:

$$q(s_i, s_j) = \sum_{l \in \{F, In, Out, P, \mathrm{E}\}} \omega_l \times q(l(s_i), l(s_j)). \quad (4)$$

Where $\omega_l \in [0,1]$ is the weight assigned to the l^{th} service description attribute and $\sum_{l \in \{F, In, Out, P, \mathrm{E}\}} \omega_l = 1$. In this way preferences on quality one some desired service attribute can be done by simply adjusting ω_l e.g., the functional category of a service could be weighted higher. Finally, the results returned by (4) is a pair of values in $(0,1] \times (0,1]$ referring to the common description rate between service s_1 and s_2.

The quality of semantic similarity between services can be then compared by analysing q i.e., their q_{cd} elements. For instance $q(s_i, s_j) > q(s_i, s_k)$, if the common description rate of $q(s_i, s_j)$ is higher than $q(s_i, s_k)$.

3.4.2.2 Context Similarity

3.4.2.2.1 Context Modeling

Besides the semantic-based functional description of web services, context information is required. Several contexts are given as mentioned before. When a user downloads a web service, he will be asked to choose his situation under these contexts provided. Context information then is modeled as a hierarchy tree which can help aggregating ratings in the sparse situation. When the data are very sparse under one context, we assume that the prediction is more accurate by using the data under similar contexts than using those under remote contexts. This assumption is detailed explained and proved in [26]. The hierarchy tree of the contexts is modeled as below:

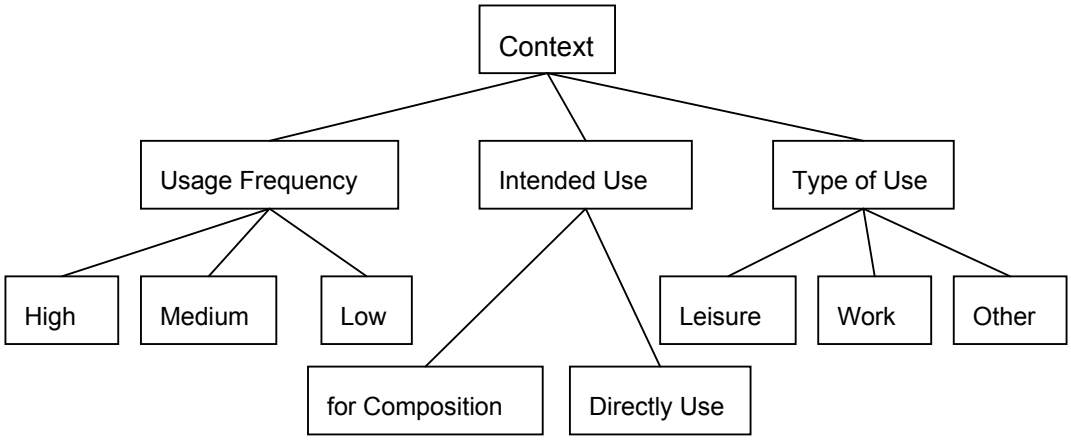

Figure 6 - Context Taxonomy

In order to find similar contexts, the similarity between two contexts needs to be computed. Two situations are covered in the following. The first one is within one context, the similarity between any context dimensions, the other one is cross contexts, the similarity between any two context cells. In this work, a context dimension is one of the contexts, such as usage frequency or intended use. A context cell includes the values cross context dimensions, for example, High Usage Frequency, Used for Composition, and work is the usage purpose, these three values together is seen as a context cell.

3.4.2.2.2 Single Context Dimension

In general, contexts are classified into three types, *scale*, *ordinal* and *categorical* in [26][21]. Different types have their own similarity computation ways. As in our work, only categorical context type is involved, thus we use semantic context similarity here.

First of all, context needs to be modeled in a DL ontology. The similarity between two context values under each context dimension C will be computed by measuring the semantic similarity from an active context value c_j to a substitute context value c_j' using their Common Description rate, as described above, using equation (3). It also can be simplified as:

$$q(c_j, c_j') := \frac{|lcs(c_j, c_j')|}{|H| + |lcs(c_j, c_j')|} \quad (5)$$

wherein the Extra Description H is a solution of the Concept Abduction problem $\langle L, C, D, O \rangle$, as $sd_j \setminus sd_i$ in equation (3). Once the semantic similarity is measured, the similarity of categorical context types converts to numerical data, formula (5) can be used for predicting ratings of active categorical context.

3.4.2.2.3 Cross Context Dimension

After getting the similarities between any two context values within one context, then the similarities between any two context cells can be computed.

The distance between two context cells E and E' can be computed through Euclidean Distance as below:

$$sim(E, E') = \frac{1}{N} \sqrt{\sum_{i=1}^{N} q(c_j, c_j')^2} \quad (6)$$

N is the number of values in the context cell, while i is the context dimension, and

$q(c_j, c_j')$ is the similarity between two context values c_j and c_j' under context dimension i.

3.4.2.3 Proposed Approach

In the case of insufficient user feedback, semantic content-based approach will be applied to solve this 'cold-start' problem (when there is lack of rating data, recommendation cannot be made before a considerable history has been collected). When the user is clicking around to search for specific web services, he will be asked for the intended web service execution environment, which is the context information we presented above. Then the system collects his viewing history, and recommends him the services under the context he has provided. If there are very few services under that context, then the web services under its similar context will be recommended. Detailed are described in the following.

Stage 1. Semantic Content-based Approach. Starting from the cold start problem, our content-based approach is considered based on the semantic representation of web services. First of all, services are described along their semantic attributes, as mentioned in Section 3.4.1.2 i.e., at functional level. Then the similarity of any two semantic described web services can be computed through formulas (3) and (4), and the detailed process is described in Section 3.4.2.1. When the user clicks to view one service, he will be asked to input his context information by selecting from the given context values. Then the top N of its most similar services under this specific context can be listed.

Based on this initial stage recommendation, we then collect users' interactions both implicitly and explicitly. One of the advantages of this approach is the semantic similarity between one web service and other services are computed offline, which can save lots of time for the real-time recommendation.

Stage 2. Context Segments. With the assumption that the prediction for data under a specific context is more accurate by using its similar context than remote context, our first choice is do prediction under its own context. We first group data under each context cell. And then when the user provides his context information, we match it with our context cell, and then compute the similarity of web services under this context cell. If there are not enough web services, then we use data under the similar context. The similarity between context cells E and E' are computed through equations (5) and (6). Then the services similarity will be computed as:

$$q(s_i, s_j) \square\ sim(E, E') \times \sum_{l \in \{F, In, Out, P, \mathrm{E}\}} \omega_l \times q(l(s_i), l(s_j)) \qquad (7)$$

To initialize context usage, we need to some knowledge to suggest which context cells the web services belong to. These tabs can be changed with viewing the users' histories.

Stage 3: Recommendation Generation. The recommendation generated based on the assumption that the most similar web services will get more chance to be viewed or downloaded. Top N services with highest similarities are recommending to the user. All users' download histories are stored with their context for further grouping web services into their context cells.

4. Architecture, Installation and Configuration

This section illustrates how to install and configure the RS Component. Since the software is composed by a set of subsystems, each with its peculiarities and requirements, we firstly explain the general software architecture and the installation of the RS Component within the Consumption Platform and then we give the instructions to complement them with the specific recommenders explained in Chapter 3.

It is worth noting that we do not provide any test dataset together with the software. The data needed to use this component shall be provided or generated in the global SOA4All system; for example, the logs needed by the collaborative filtering approach are automatically created by the interaction of the users with the SOA4All Consumption Platform.

4.1 Architecture of the whole Recommender System and installation of the RS Component

In Section 3.1 we explained that our Recommender System is a hybrid recommender system with a set of recommenders orchestrated by an additional component that integrates the results of each of them.

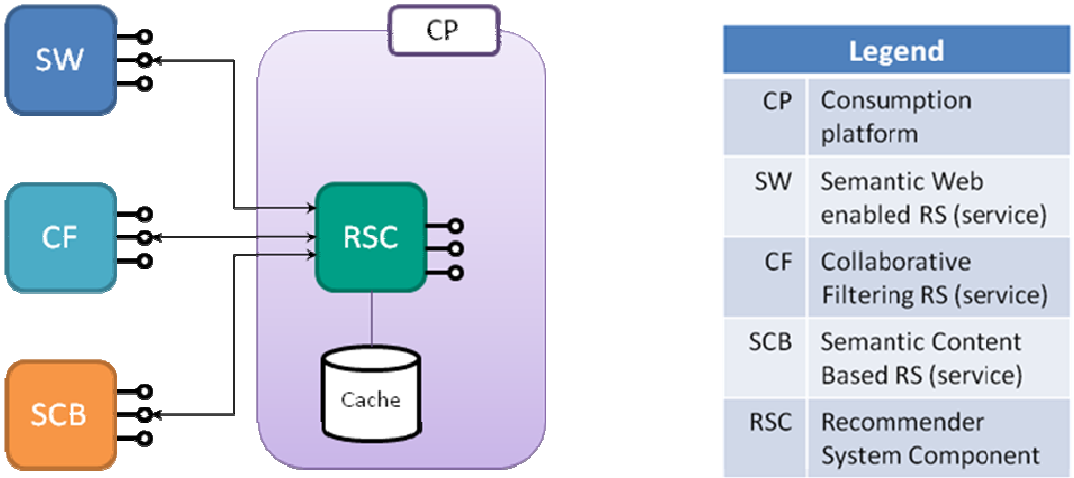

Figure 7 – Recommender System Architecture and its integration in SOA4All

In Figure 7 we represent the Recommender System by an architectural and deployment point of view: each recommendation technique was exposed as a REST service. Each service provides an interface with methods similar to the RS API described in Section 2.2 (it means that it's possible to invoke the services for both batch time and run time tasks).

The orchestration of the recommenders is done by the Recommender System Component (RSC), a library used to supply the Recommender System features hiding the complexity of the inner components.

The figure shows also how the Recommender System works in SOA4All: while the three REST services are exposed on remote locations, the RSC library is used internally by the Consumption Platform in order to interact with the Recommender System, e.g. asking for recommendations.

The instructions included in this section are also available on line on the project wiki at http://soa4all-wp1.sti2.at/index.php/RecommenderSystem; that page will be updated whenever needed, thus the interested reader is suggested to take the wiki as reference for installation and configuration of the RS Component.

4.2 Installation and configuration of the RS Component

The RS Component consists of a library to be used by the Consumption Platform. The jar file is available on Nexus repository at:
http://coconut.tie.nl:8080/nexus-webapp-1.3.1/content/repositories/3rdparty/it/cefriel/swa/rs/

It also has a configuration file, to be filled with the correct locations of the internal local and remote components as explained above and depicted in Figure 7.

Requirements for a complete installation of the RS Component are:

- **Java JDK 1.5** or greater.
- **MySQL Server 5.1** or greater. For installation instruction, see http://dev.mysql.com/doc/refman/5.1/en/installing.html.

4.3 Installation and configuration of the Collaborative Filtering Recommender System

The installation instructions were already available in deliverable D2.7.1 [1].

4.4 Installation and configuration of the Semantic Web enabled Knowledge-based Recommender System

As reported above, the Semantic Web enabled Recommender System is built over the LarKC platform [13][14], an application able to run sequences of plug-ins, named *workflows*; a plug-in is a software component that can execute some tasks.

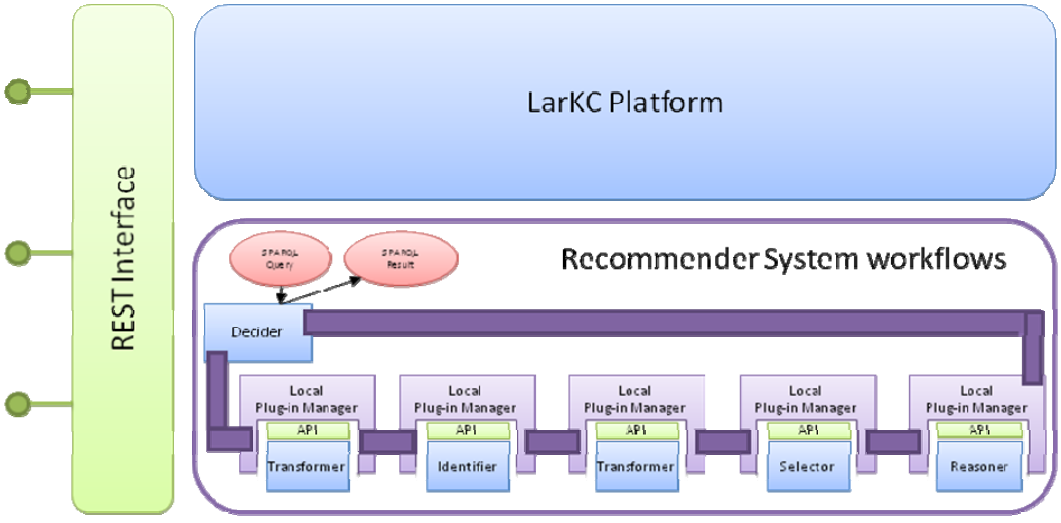

Figure 8 – Semantic Web enabled Recommender System architecture

In order to install the Semantic Web enabled Recommender System the following components are required (see Figure 8):

- the LarKC platform v1.0, available at http://sourceforge.net/projects/larkc/. The installation instruction are available in the same Web site [14];
- the plug-ins required in the workflows that compute the recommendations are available at

a dedicated SourceForge project https://sourceforge.net/projects/larkc-answers/. The instruction to deploy them are available in the LarKC platform documentation [14];

- the REST interface of the Semantic Web enabled Recommender System is available at the same dedicated SourceForge project https://sourceforge.net/projects/larkc-answers/ (with the instruction for the configuration). The service is a Java web application and it should be deployed in a servlet container like Apache Tomcat.

Please note that in order to work this recommender system requires an available Web connection to retrieve all the required data: the users/services from the Semantic Spaces, the services' descriptions from iServe and the Analysis Platform and so on.

4.5 Installation and configuration of the Semantic Content-based Recommender System

The code of the Semantic Content-based Recommender System is on the SOA4All project SVN versioning system at https://svn.sti2.at/soa4all/trunk/soa4all-studio-decoupled/content-based-service-recommendation.

In order to interact with the Semantic Content-based Recommender System the following components are required:

- a pool of (SA-WSDL) semantic-based services (based on the Minimal Service Model - http://cms-wg.sti2.org/TR/d12/v0.1/) are stored in a RDF repository [31]. Their descriptions need to be based on an ALE TBox in order to evaluate semantic similarity between services.
- a Semantic Reasoning module (DL reasoner Fact++ [32]) responsible for specific DL inferences such as subsumption (e.g., matching quality), difference (Common description rate).

5. Conclusions

This document illustrated the RS component developed for the SOA4All Studio. It reported about the architecture and API of the component and its relation with other SOA4All components, it detailed the different algorithms and techniques employed and the instruction on how to use the component.

The component is integrated within the SOA4All Studio and leverages the information provided by the other SOA4All components, like service semantic descriptions, user profiles, logs of user interactions with the platform and of service execution, service monitoring information, etc. The component also makes use of external linked data sources to complement the internal data with further details to improve recommendations.

The component offers its functionalities to the SOA4All Studio via a well-defined API which enables the display of service recommendations to the SOA4All users in a dedicated “box” which lists the RS component suggestions together with a score and/or a recommendation explanation when available.

This second prototype of the RS component not only shows the implementation progresses with regards to the first version described in D2.7.1, but it also demonstrates the advances in the realization of novel and beyond state-of-the-art recommenders and their integration into a unique and comprehensive Hybrid Recommender System.

This deliverable is complemented by a set of papers accepted (or still under review at the time of writing) at major conferences, both in the area of Recommender Systems and in the field of Semantic Systems.

References

[1] Guillermo Álvaro Rey, Dario Cerizza, Giovanni Di Matteo, Gianluca Ripa, Andrea Turati, Matteo Villa: "D2.7.1 – Recommender System First Prototype", SOA4All project deliverable, available at http://www.soa4all.eu/file-upload.html?func=startdown&id=142, August 2009.

[2] Andrea Turati, Dario Cerizza, Irene Celino and Emanuele Della Valle: "A Collaborative Filtering System for Recommending Web Services through the Analysis of User Actions within a Web 2.0 Portal", In Proceedings of the 3rd Workshop on Web Personalization, Reputation and Recommender Systems (WPRRS 2009) at the 2009 IEEE/WIC/ACM International Conference on Web Intelligence, Milano, Italy, September 2009.

[3] Daniele Dell'Aglio, Irene Celino, Dario Cerizza, Emanuele Della Valle, Andrea Turati: "D5.3 – User and Service Clustering Research Report – Version 2" Service-Finder project deliverable, available at http://www.service-finder.eu/attachments/D5.3.pdf, December 2009.

[4] Saartje Brockmans, Irene Celino, Dario Cerizza, Daniele Dell'Aglio, Emanuele Della Valle, Michael Erdmann, Adam Funk, Holger Lausen, Nathalie Steinmetz: "D7.5 – Final Report on Assessment of Tests for Beta Release", Service-Finder project deliverable, available at http://www.service-finder.eu/attachments/D7.5.pdf, December 2009.

[5] Robin Burke: "Knowledge-Based Recommender Systems". Encyclopedia of Library and Information Science, 69(32), 2000.

[6] Eyal Oren, Ronald Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovanni Tummarello: "Sindice.com: a document-oriented lookup index for open linked data". International Journal of Metadata, Semantics and Ontologies, 3(1):37-52, 2008.

[7] Mathieu d'Aquin, Claudio Baldassarre, Laurian Gridinoc, Sofia Angeletou, Marta Sabou, and Enrico Motta. "Characterizing Knowledge on the Semantic Web with Watson". In Proceedings of the 5th International Workshop on Evaluation of Ontologies and Ontology-based Tools (EON2007), co-located with the ISWC2007, pages 1-10, Busan, Korea, 2007.

[8] Chris Bizer, Jens Lehmann, Georgi Kobilarov, Soren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. "DBpedia – A Crystallization Point for the Web of Data". Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 7:154-165, 2009.

[9] Dan Brickley and Libby Miller. "FOAF Vocabulary Specification 0.97". Available on line at http://xmlns.com/foaf/spec/, January 1st, 2010.

[10] Andy Seaborne and Eric Prud'hommeaux. "SPARQL Query Language for RDF - W3C Recommendation". Available at http://www.w3.org/TR/rdf-sparql-query/, January 15th, 2008.

[11] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. "SWRL: A Semantic Web Rule Language Combining OWL and RuleML". Available on line at http://www.w3.org/Submission/SWRL/, 2004.

[12] Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres, and Dave Reynold. "RIF Core Dialect - W3C Recommendation". Available on line at http://www.w3.org/TR/rif-core/, June 22th, 2010.

[13] Dieter Fensel, Frank van Harmelen, Bo Andersson, Paul Brennan, Hamish Cunningham, Emanuele Della Valle, et al. "Towards LarKC: A Platform for Web-Scale

Reasoning", In: Proceedings of the 2008 IEEE international Conference on Semantic Computing ICSC, pp. 524--529, IEEE Computer Society (2008).

[14] Georgina Gallizo, Alexey Cheptsov, Matthias Assel, Luka Bradesko, Vassil Momtchev: "LarKC Platform Manual – V1.0", April 2010, Available on line at http://sourceforge.net/projects/larkc/files/Release-1.0/LarKC_PlatformManual_V1_0.pdf.

[15] Carlos Pedrinaci, Dong Liu, M. Maleshkova, David Lambert, Jacek Kopecky and John Domingue: "iServe: a Linked Services Publishing Platform", Workshop: Ontology Repositories and Editors for the Semantic Web at 7th Extended Semantic Web Conference (2010).

[16] Robin Burke: "Hybrid Recommender Systems: Survey and Experiments". User Modeling and User-Adapted Interaction, 12(4):331-370, 2002.

[17] Xiaoyuan Su and Taghi M. Khoshgoftaar: "A Survey of Collaborative Filtering Techniques". Advances in Artificial Intelligence, vol. 2009(Article ID 421425), 2009.

[18] Michael J. Pazzani and Daniel Billsus: "Content-Based Recommendation Systems". The Adaptive Web, pages 325-341, 2007.

[19] Baader, F. and W. Nutt (2003). The Description Logic Handbook: Theory, Implementation, and Applications.

[20] Brandt, S., R. Küsters, et al. (2002). Approximation and difference in description logics. KR: 203-214.

[21] Chen, A. (2005). Context-Aware Collaborative Filtering System: Predicting the User's Preference in the Ubiquitous Computing Environment. Location- and Context-Awareness, Springer Berlin / Heidelberg. 3479/2005: 244-253.

[22] Cohen, W. W., A. Borgida, et al. (1992). Computing Least Common Subsumers in Description Logics. AAAI: 754-760.

[23] Küsters, R. (2001). Non-Standard Inferences in Description Logics, Springer.

[24] Lecue, F. and A. Delteil (2007). Making the Difference in Semantic Web Service Composition. AAAI: 1383-1388.

[25] Li, L. and I. Horrocks (2003). A Software Framework for Matchmaking Based on Semantic Web Technology. WWW: 331-339.

[26] Liu, L., F. Lecue, et al. (2010). Using Context Similarity for Service Recommendation. Fourth IEEE International Conference on Semantic Computing.

[27] Maamar, Z., S. K. Mostefaoui, et al. (2005). Context for Personalized Web Services. Proceedings of the 38th Hawaii International Conference on System Sciences.

[28] Mostefaoui, S. K. and B. Hirsbrunner (2004). Context Aware Service Provisioning. IEEE/ACS International Conference on Pervasive Services (ICPS'04): 71-80.

[29] Noia, T. D., E. D. Sciascio, et al. (2003). A System for Principled Matchmaking in an Electronic Marketplace. WWW: 321-330.

[30] Paolucci, M., T. Kawamura, et al. (2002). Semantic Matching of Web Services Capabilities. ISWC: 333-347.

[31] C. Pedrinaci, D. Lambert, M. Maleshkova, D. Liu, J. Domingue, and R. Krummenacher, "Adaptive Service Binding with Lightweight Semantic Web Services", 2010, ch. Service Engineering: European Research Results.

[32] I. Horrocks, "Using an expressive description logic: Fact or fiction?" in KR, 1998, pp. 636–649.

Annex A. List of papers

In this annex, we list the relevant scientific papers concerning the components illustrated in this document. For the interested readers, the accepted papers are attached to the deliverable.

Accepted papers:

- Andrea Turati, Dario Cerizza, Irene Celino and Emanuele Della Valle: **"A Collaborative Filtering System for Recommending Web Services through the Analysis of User Actions within a Web 2.0 Portal"**, In Proceedings of the 3rd Workshop on Web Personalization, Reputation and Recommender Systems (WPRRS 2009) at the 2009 IEEE/WIC/ACM International Conference on Web Intelligence, Milano, Italy, September 2009.
- Liwei Liu, Freddy Lecue, Nikolay Mehandjiev and Ling Xu: "**Using Context Similarity for Service Recommendation**", in Proceedings of the Fourth IEEE International Conference on Semantic Computing (ICSC 2010), September 2010.
- Freddy Lecue: "**Combining Collaborative Filtering and Semantic Content-based Approaches to Recommend Web Services**", in Proceedings of the Fourth IEEE International Conference on Semantic Computing (ICSC 2010), September 2010
- Liwei Liu, Nikolay Mehandjiev and Ling Xu: "**Using Contextual Information for Service Recommendation**", in Proceedings of Hawaii International Conference on System Sciences-44 (HCISS-44), January 2011.

Papers submitted but still under review at the time of writing:

- Daniele Dell'Aglio, Irene Celino: "**anSWERS – a novel Semantic Web-enabled Recommender System**", Submitted to the 9th International Semantic Web Conference 2010.
- Liwei Liu, Freddy Lecue and Nikolay Mehandjiev: "**A Hybrid Approach for Software Service Recommendation**", Submitted in ECOWS 2010.

A Collaborative Filtering System for Recommending Web Services through the Analysis of User Actions within a Web 2.0 Portal

Andrea Turati, Dario Cerizza, Irene Celino
CEFRIEL – ICT Institute Politecnico di Milano
Via Fucini 2, 20133 Milano, Italy
{*andrea.turati, dario.cerizza, irene.celino*}*@cefriel.it*

Emanuele Della Valle
Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{*emanuele.dellavalle*}*@polimi.it*

Abstract

The current Web manifests the problem of information overload, especially due to the success of the Web 2.0 paradigm, in which users provide new contents quickly. To help people to find the most valuable information, many Web sites includes a recommendation system based on a rating mechanism. However, such approach cannot be used when a rating mechanism is not present and, in addition, it does not take into account all the actions performed by the users. We propose an extension of the collaborative filtering approach to design a more effective recommendation system that overcomes those limitations.

1. Introduction

The term *information explosion* describes the rapidly increasing amount of published information. Many persons use that term to describe the current situation of the Internet. Indeed, every day new data appear on the Web, especially due to the proliferation of blogs, wikis and the so called *social communities* in which people can share photos, comments and other contents. The "information explosion" can lead to the "information overload", that is the situation where there is far too much information at people disposal so that useful information could be "hidden from view" by other data. Thus, techniques to retrieve *useful* information become more and more important.

A special kind of information retrieval techniques that focuses on this issue is named *information filtering* [15]. As the name suggests, starting from a big set of information this technique identifies a small subset which should include the useful/interesting information (i.e. it discards redundant, unwanted or irrelevant information).

Information filtering is applied to many areas and many tools implementing such techniques exist (e.g. anti-SPAM systems). Recommendation systems are a specific type of information filtering technique that attempts to present information items (e.g. movies, songs, books, news, images, Web pages) that are likely of interest to the user.

Recommendation systems come from the observation that people tend to ask friends for advises or to read/listen to expert help when presented with a number of unfamiliar alternatives [25]. The first recommendation systems appeared at the end of 1990s [8], [18], [24] and since then it has been an active research area both in the industry and academia. Many examples of such applications were developed [22] by Amazon.com [11], MovieLens [12], NetFlix [2], Pandora and Last.fm [9], and many others.

The majority of the successful Web sites including recommendation systems usually implements a mechanism that allows users to explicitly assign ratings to the items. However, this method cannot be adopted in context where users are not allowed to leave an explicit evidence about their preference. Furthermore, even if such a mechanism exists, this approach does not take into account many aspects of the user behavior that might be relevant.

In this paper we inspect how it is possible to analyze the users' behavior in order to improve the quality of the recommendations and we describe a project named *ServiceFinder* [5] where we implemented the approach presented in this paper, which we aim to extend in the *SOA4All* project[1].

Section 2 provides a brief description of the current state of the art of the recommendation systems. Section 3 proposes an approach that enriches the "standard" collaborative filtering technique in order to improve the quality of the recommendations by trying to understand the real preferences of the users. It also describes a real use-case where such approach has been implemented, focusing on the architecture of the component responsible for making recommendations. The paper ends with a description of some ways that can be followed to improve the approach.

2. State of the Art

Information filtering systems decide to select or discard items taking into account the user that will get the results. This is done by comparing the user's *profile* (i.e. a representation of his interests or tastes) to some reference character-

1. http://www.soa4all.eu/

istics. A user's profile can be created and maintained either explicitly (i.e. the user specifies it by stating his preferences) or implicitly (i.e. the system monitors his behavior and makes deductions). The characteristics that are compared to the user's profile depend on the algorithm implemented into the information filtering system (or the recommendation system). In literature, two different approaches exist [1]: the *content-based* approach extracts such characteristics directly from the information items while the *collaborative filtering* approach derives them from the user's social environment.

Given a user, a content-based recommendation system suggests those items having the highest correlation between their contents and the user's profile (i.e. the user will be recommended items similar to the ones the user preferred in the past). Given a user, a collaborative recommendation system suggests those items preferred by people with a profile most similar to the user's one (i.e. the user will be recommended items that people with similar tastes and preferences liked in the past). In the following sections we describe the two approaches.

2.1. Content-based Filtering

From the assumption that users that liked certain items will like similar items too, such algorithms compare pairs of items in order to understand their similarities. Therefore, given an item that a user liked very much in the past (i.e. the user assigned it a high rating), they can suggest the most similar items being confident that the user will like it.

The item-to-item similarity is computed comparing their contents or properties ([22] calls this approach "item-to-item correlation"). For example, in a music application, in order to recommend songs to a given user, the similarities among the songs the user prefers are evaluated by inspecting their features (e.g. artists, genres, etc), and then the songs that have a high degree of similarity to the user's preferences are recommended. Therefore, a content-based recommendation system learns a profile of the user's interests based on the features present in items the user has rated.

There exist several ways to compare user's preferences to item features. A technique represents both user preferences and item features by means of vectors in the same multidimensional space (e.g. through the *term frequency indexing* [19]) and uses the cosine similarity measure [13] as an estimation of the probability that an item is liked by the user. Another technique exploits Boolean indexes [4]. Other techniques implement probabilistic approaches [7], like Bayesian classifiers [17], [14], natural language algorithms [10], decision trees, artificial neural networks and many other.

2.2. Collaborative Filtering

Collaborative filtering aims to learn user preferences and make recommendations based on user and community data. It assumes that every user rates some items to reflect his satisfaction about them. The item ratings of a user represent the user profile, which consists of a set of items associated with a value reflecting the user opinion about them. Then, user profiles are compared in order to identify groups of similar user profiles. Such user profile clusters are used to come up with recommendations: given a user, the system may suggest to him those items that he has not yet seen while other users – whose profiles belong to the same cluster of the given user profile – appreciated a lot.

Mathematically, the problem can be modeled through a matrix, where users and items intersect. Let U be the set of all n users that use the system and I the set of all m items managed by the system. A generic user $u_j \in U$ can express his opinion about an item $i_k \in I$ by assigning a rating r_{u_j,i_k}, which is normally in a binary or numerical scale. Thus, a matrix R containing ratings can be sketched as in equation 1.

$$R = \begin{bmatrix} r_{u_1,i_1} & r_{u_1,i_2} & \cdots & r_{u_1,i_m} \\ r_{u_2,i_1} & r_{u_2,i_2} & \cdots & r_{u_2,i_m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{u_n,i_1} & r_{u_n,i_2} & \cdots & r_{u_n,i_m} \end{bmatrix} \quad (1)$$

The ratings that user u_j assigned to items represent his preferences, so the user profile used for computing recommendations can be formulated as $\{r_{u_j,i} | i \in I\}$, which is the row corresponding to user u_j in the matrix R. Analogously, a column of the matrix R corresponds to an item and contains the ratings that users gave to that item – it is a sort of "item profile" which reflects the overall satisfaction on that item.

Accordingly to [3], there are two general classes of collaborative filtering algorithms, which differ in the use of the matrix R: memory-based and model-based. Both make recommendations by computing rating predictions: they estimate the ratings that the given user would assign to the items he has not yet seen and suggest him the ones with the highest estimated ratings. The following sections provide an overview of those approaches.

2.2.1. Memory-based Collaborative Filtering. These algorithms are heuristics that make rating predictions based on the entire collection of the rated items. Basically, through the analysis of the matrix R, the rating that a given user would assign to a specific item is estimated taking into consideration the user similarities computed before.

The similarity between two users is computed by comparing their profiles. Given a user the algorithm identifies his profile (i.e. a row in the matrix R) and compares it with all the other user profiles (i.e. other rows of the matrix R). In this way, user profiles that are most similar to the given one (i.e. their corresponding two rows in the matrix contain similar values, which means that the users gave similar ratings to the same items) are marked as *neighbors*

of the given user. Finally, the ratings of the neighbors are used to estimate the rating that the target user would give to a specific unseen item.

One of the most used equation to measure the similarity between users is the Pearson correlation coefficient [18], [24]. Another one is the cosine-based approach [20], [3], where two users are represented by two different vectors in a multidimensional space and their similarity is computed as the cosine of the angle between them. Another method is to use the mean squared difference [24].

A variation of the memory-based approach described so far has been named *item-based* or *item-to-item* collaborative filtering [11], [20], [6]. The difference with the method described above is the objects of the similarity measure. In the standard memory-based approach the similarity is calculated for each pair of users, while in the item-based approach the similarity is calculated for the items. In other words, the former compares the rows of the matrix R, while the latter compares its columns.

Rather than matching the user to other similar users, item-based collaborative filtering matches each of the user's rated items to other "related" items. Such "relatedness" between two items reflects the fact that those two items have been consumed together or rated equally.

2.2.2. Model-based Collaborative Filtering. This approach aims at compiling a mathematical model reflecting user preferences. This can be done by first compiling (off-line) the complete data set into a descriptive model of users, items and ratings and then computing recommendations by consulting the model.

Early research on this approach evaluated two probabilistic models: *Bayesian clustering* and *Bayesian networks* [3]. These models are used to estimate the probability that a user will give a particular rating to an item given the user's ratings of the previously rated items.

They use the matrix R to learn some internal parameters and exploit clustering techniques to group the users. Due to internal limitations, they do not work well in domains where users assume different positions because they cannot cluster a user into several categories at once.

Many other model-based collaborative filtering approaches appeared in the literature: statistical models based on standard algorithms coming from data mining (e.g. K-means) [26], linear regressions [20], entropy models [16], stochastic techniques like Markov decision processes [23], rule-based approaches [21] and many other.

3. Designing a Recommendation System for a Web 2.0 portal

Service-Finder is a portal[2] that allows users to search for Web services. It adopts the Web 2.0 paradigm, where users can add tags, assign ratings, manage their bookmarks and so on. Beside the three standard search functionalities (i.e. free-text search string, a category tree and a tag cloud), it suggests Web services through a recommendation system. SOA4All is another project that goes beyond Service-Finder by providing users with the possibility to semantically describe and execute both services and processes that involve them. It also includes a recommendation system to suggest semantically annotated services and other entities.

In this section we provide an overview of the the way that we followed in Service-Finder (and that we are going to extend in SOA4All) to design and implement a system that exploit rich information to make recommendations. In particular, Section 3.1 describes how to exploit the whole information coming from a Web 2.0 portal to improve the quality of recommendations, with a special focus on Service-Finder. Section 3.2 shows the architecture of the component implementing the approach described before in the context of Service-Finder.

3.1. The Approach for Making Recommendations

Every approach described in Section 2 assumes to have a set of users, a set of items and the ratings that users assigned to items, and uses them in order to identify several items that a given user might appreciate. In the context of Service-Finder, from the point of view of the recommendation process the services that users can browse through represent the so-called "items".

In addition, Service-Finder allows users to rate services. The ratings that users assign to services can be used by the recommendation process to evaluate user profiles and make recommendations. However, we believe that the quality of recommendations would be improved by taking into account other information rather than only the ratings. It is true that a rating explicitly reflects the opinion of a user about a service, but the fact that a user prefers a service can be inferred by the observation of the user behavior with respect to the service. Indeed, within the Service-Finder portal users perform many actions with respect to the services – and not only assigning ratings – and, for example, if a user views the details of a specific service many times and spends a lot of time in editing the service details, then it is possible to say with confidence that the user is addicted to that service.

While interacting with the Service-Finder portal, users perform many actions: they select some services from the search results, they rate services, they insert services into their bookmarks, they view some service details, they try to invoke some service operations, and so on. In general, we can say that every user establishes some kind of relation with a set of services.

Comparing the relations relating to two different users, it is possible to estimate the degree of similarity between the two users: if the set of services tight connected to a user

2. http://demo.service-finder.eu/

overlaps considerably the set of services tight connected to the other one, then the two users are somehow similar. In this way, similarities between all users are computed. Then, given a specific user, it is possible to suggest services that he might be interested in, because those services are appreciated by other users that are similar to him.

The recommendation process is analogous to the ones already described in Section 2, except for the relations between users and items. In literature users and items are related by means of ratings, while here users and services are related by means of generic relations, whose strength is evaluated by inspecting actions that users perform.

Not all actions can be used in evidence for increasing the user opinion about the service involved. Furthermore, not all actions have the same importance in establishing the relation between user and service. For this reason, we decided to associate a weight to each action that a user may perform while visiting the Service-Finder portal. By properly combining all weighted actions that a user did related to a specific service, the system identifies a number reflecting the strength of the relation between the user and the service. A strong relation between a user and a service means that the user really appreciates the service and would recommend it. A weak relation means that the user is not interested in the service or dislikes it, so it is not worth recommending it. The numbers representing the strength of the relations represent the values included in the user-item matrix described in Section 2, which a standard recommendation system (i.e. coming from the literature) uses to make recommendations.

Table 1 lists all the actions that a user may perform on a service: the first one is the most relevant action in establishing a connection between the user and the service while the last one has the lowest impact on that connection. The table also shows the actions that have a negative influence on relations between users and services. This means that the strength of the relation between a user and a service is reduced whenever one of those actions occurs.

Action	Weight
Assign a high/positive rating with a comment to a service	10
Assign a high/positive rating without a comment to a service	8
Add into bookmarks	7
Assign a tag to a service	6
Edit a service	5
Try to invoke a service	4
Click a link related to a service to go to an external document	3
Compare a set of services	3
View the details of a service	2
Select a service (e.g. from the search results)	1
Remove the service from bookmarks	-2
Assign a low/negative rating with a comment to service	-5
Assign a low/negative rating without a comment to service	-10

Table 1. The initial weights of the user actions

There is no rule that can be used to assign the right values of such weights, because they depend on the context where the system is used. Initially, we set the weights listed in Table 1. Higher positive weights are associated to actions giving a clear evidence of the high appreciation of the item, while lower positive weights are associated to actions where the appreciation is not so evident or is lower. On the other hand, the actions giving a clear evidence of the rejection of the item are associated with negative weights.

We are conscious that the weights are fundamental to produce significant recommendations, so we are going to fine-tune them based on the inspection of the goodness of the retrieved recommendations.

3.2. Architecture and Implementation of the Recommendation Component

Figure 1 shows the internal architecture of the recommendation component (the arrows represents data flow). The *Parser* is responsible for getting log files produced by the Service-Finder portal and extracting information from them. The extracted information is temporarily inserted into a database, named *User History*, which represents the history of all user actions done in the portal. In addition, the *User-Service Correlation Analyzer* accesses the information stored in the User History and calculates the relations between users and services. The result of the analysis is a matrix, named *User-Service Matrix*, where each row represents a user and each column represents a service; the cell where a service intersects with a user stores a value that represents how much that service is related to the user[3]. Starting from the matrix that contains values representing the intensity of the relations between users and services, the *Recommender* makes recommendations, that is given a user it returns a list of recommended services.

The tasks performed by the whole component can be represented as two distinct conceptual parts. One is responsible for information extraction and analysis and the other is responsible for making recommendations. The former task is the result of the batch execution of Parser and User-Service Correlation Analyzer, while the latter is done on-line by the Recommender.

The two conceptual tasks run independently. From the implementation point of view, the first conceptual task is executed as a single thread in which the execution of Parser and User-Service Correlation Analyzer are properly synchronized.

The synchronization between those components is needed to avoid typical problems that arise when different components work on the same data structure. One problem might be due to the concurrent execution of both User-Service Correlation Analyzer and Parser, which continuously

3. The User-Service Matrix has the same structure of the matrix than standard recommendation algorithms use as input (i.e. equal to the one depicted in Equation 1).

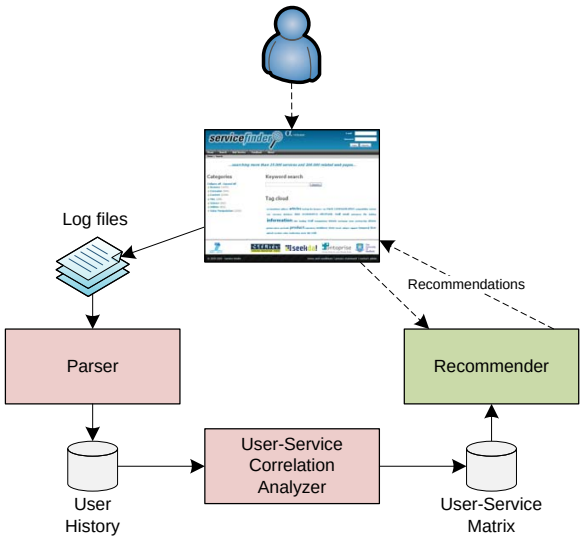

Figure 1. The internal architecture

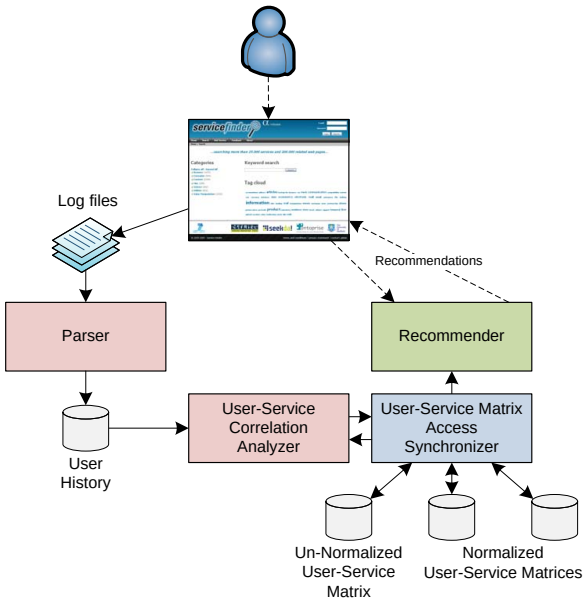

Figure 2. The refined internal architecture

monitors the file system and reacts after the appearance of a new log file. In that situation, the User-Service Correlation Analyzer would need to repeatedly retrieve data from the User History during its internal computation, while at the same time the Parser would need to insert new data into the same database (e.g. because a new log file is available). In that case, the computation of the User-Service Correlation Analyzer might encounter some problems due to unexpected new data in the database. This issue can be solved by running a single thread in which the two components are executed one by one.

Since on the one hand it takes a while for User-Service Correlation Analyzer to update all the matrix cells and on the other hand the Recommender has to provide recommendations on demand, another synchronization problem might arise due to concurrent accesses to the User-Service Matrix on behalf of both User-Service Correlation Analyzer and Recommender. For this reason, we use two copies of the User-Service Matrix. One matrix is used by Recommender to provide on-line recommendations while the other one is updated by User-Service Correlation Analyzer. When the User-Service Correlation Analyzer finishes to update the off-line matrix, then the two matrices are swapped: the one that was on-line (and was used by Recommender) becomes off-line (ready to be used by User-Service Correlation Analyzer) and the one that was off-line (just updated by the User-Service Correlation Analyzer) becomes on-line (ready to be used by Recommender). After each swap, the updates made on the "new" on-line matrix are copied into the "new" off-line matrix, to keep their content consistent.

For this reason, we add a new component named *User-Service Matrix Access Synchronizer* to the architecture which is responsible for granting access to the right matrix. In Figure 2, a third matrix also appears beside the two copies of the User-Service Matrix: it differs from them by the fact that it is not normalized (this issue will be discussed later).

In the following sections we detail the implementation of the core of our approach in details. For the sake of the simplicity, henceforth we do not distinguish between the two copies of the User-Service Matrix.

3.2.1. User-Service Correlation Analyzer. This component is responsible for estimating the strength of the relations between users and services, which means to fill in the matrix that expresses user interests for the services. In other words, it examines the User History in order to produce the User-Service Matrix (see Figure 2).

User-Service Correlation Analyzer considers every user separately. Firstly, it collects all actions made by a specific user. Then, for each action, it gets the reference to the service related to that action and updates the cell of the matrix that corresponds to the intersection between the row of the user and the column of the service by adding the action weight.

Formally, the User-Service Correlation Analyzer runs the algorithm in Figure 3, which is described using a pseudo-code. R represents the User-Service Matrix, which at the beginning is initialized with null values. $R(u, s)$ represents the cell of the matrix R corresponding to user u and service s. $max(u)$ and $min(u)$ are respectively the maximum and minimum values in the relations between user u and all services, while MAX and MIN are respectively the maximum and minimum value used to express the strength of any relations between users and services in the matrix R. Given an action a, $a.weight$ denotes the weight of the action, $a.timestamp$ denotes the instant at which the action

occurred, $a.user$ identifies the user that performed the action and $a.service$ identifies the service related to the action (e.g. if a user rated a service, then the service related to that action is the one rated by the user). $lastExecutionTimestamp$ stores the instant of the last execution of this algorithm.

1: **for all** user u **do**
2: $updatedStrengths \leftarrow \emptyset$
3: $minOrMaxChanged \leftarrow false$
4: **for all** action $a \in UserHistory$ such that $a.user = u$ and $a.timestamp > lastExecutionTimestamp$ **do**
5: $s \leftarrow a.service$
6:
7: {Update the value of relation between u and s}
8: $R(u,s) \leftarrow R(u,s) + a.weight$
9: $updatedStrengths \leftarrow updatedStrengths \cup \{s\}$
10:
11: **if** $R(u,s) > max(u)$ **then**
12: $max(u) = R(u,s)$
13: $minOrMaxChanged \leftarrow true$
14: **end if**
15: **if** $R(u,s) < min(u)$ **then**
16: $min(u) = R(u,s)$
17: $minOrMaxChanged \leftarrow true$
18: **end if**
19: **end for**
20:
21: {Normalization of the values of the user's relations}
22: **if** $minOrMaxChanged$ **then**
23: **for all** service s **do**
24: **if** $R(u,s) \neq NULL$ **then**
25: $R(u,s) \leftarrow \frac{R(u,s) - min(u)}{max(u) - min(u)} \times (MAX - MIN) + MIN$
26: **end if**
27: **end for**
28: **else**
29: **for all** service $s \in updatedStrengths$ **do**
30: **if** $R(u,s) \neq NULL$ **then**
31: $R(u,s) \leftarrow \frac{R(u,s) - min(u)}{max(u) - min(u)} \times (MAX - MIN) + MIN$
32: **end if**
33: **end for**
34: **end if**
35: **end for**

Figure 3. Evaluation of the strength of the relations between users and services

The first time this algorithm is executed, the User-Service Matrix contains only null values. When the algorithm is executed, for each user it extracts from the User History all actions that the algorithm has not yet taken into consideration. Then, every action implies to update the value of a cell of the matrix: the weight of the action is summed to the value already stored in that cell. Finally, a normalization step is executed to spread the values of a user over a common numerical scale in order to make user profile (i.e. the rows of the matrix) comparable. To do that, it is necessary to keep track of both the maximum and the minimum values of the cells corresponding to the user (i.e. $max(u)$ and $min(u)$) and compare them with the maximum and minimum values allowed in the final matrix (i.e. MAX and MIN).

In the calculation of relation values the action weights are summed, so a value will be very high for the users that performed many actions related to the corresponding service, while a value will be very low for users that performed few actions related to the corresponding service. Since the users perform different numbers of actions, the values contained in the matrix will vary on different scales. To allow comparison between any users (irrespective of the number of actions they performed), it is necessary to shift the values of every user to a common scale (normalization step).

After the normalization of all values, the values in all the rows of the matrix can be used to make recommendations. However, if the algorithm is executed once more (for example, because the User-Service Matrix needs to be updated taking into account new actions performed by the users within the portal), the action weights cannot be simply summed to the value included in the cells because the value has been normalized while the weights have not.

For this reason, a User-Service Matrix containing values computed by the algorithm in Figure 3 without the normalization step is maintained and managed by the User-Service Matrix Access Synchronizer (in Figure 2 is represented as *Un-Normalized User-Service Matrix*).

3.2.2. Recommender. Recommender is responsible for making recommendations. Our implementation is based on Taste[4], an extensible framework that implements many collaborative filtering algorithms available in literature.

Firstly, it compares the rows of the user-service matrix – representing the users' profiles – in order to calculate the similarity between users. The most similar users to the given one form the user's neighborhood. At run-time, given a specific user, the profiles of his neighbors are taken into account jointly in order to identify the most appreciated services (i.e. the matrix cells containing the highest values). From that set of services, the ones that the given user has not yet seen are recommended.

4. Conclusion and Future Works

In this paper we proposed an extension of the collaborative filtering approach for making recommendations. It is

4. http://taste.sourceforge.net

designed to exploit all the information coming from a Web 2.0 portal – rather than merely the ratings explicitly assigned by users – in order to build more accurate users' profiles, so that more effective recommendations are provided. In addition, we described a real use-case (Service-Finder) where the approach has been implemented and we cited another project (SOA4All) where we are going to improve the approach. Below we outline our evaluation plan and in the following paragraphs we provide a list of some of the improvements that can be implemented in the future.

4.1. Evaluation

We would like to have both some data that demonstrate the goodness of the recommendations provided with our approach, and some guidelines that help developer in setting appropriate action weights. To evaluate our approach we are setting up a way to understand if a user appreciated a recommendation or not, through the analysis of the user feedbacks. The user feedbacks can be explicit (e.g. two buttons, "like" and "dislike") or implicit (which means that the system will analyze the log file in order to understand how much a user appreciated a recommended service, i.e. by analyzing how many and which actions the user performed related to the recommended services). To evaluate the action weights, we are setting up a closed testbed, which will allow us to execute the following steps: (i) select a small set of users with well-defined different profiles (i.e. users that performed quite different actions), (ii) pass the log files including the user actions to the recommendation component and inspect the suggested services for the previously selected users, then (iii) change or adjust the action weights and repeat the second step; at the end, compare the recommendations in order to evaluate the different choices.

4.2. Content-Based Extension for Matrix Sparsity

One of the well known problems of the collaborative filtering approach is that the user-items matrix can be very sparse [20], which reduces the quality of the recommendations. This is also valid for Service-Finder and SOA4All, where we can realistically think that a typical active user will interact with at most few hundreds of services, which represents about 1% out of the total[5]. This implies that every row of the matrix – representing a user profile – contains few values and consequently the user-service matrix is sparse.

Suppose that a given user interacts with two services. In this case, the collaborative approach fills in two cells of the row associated to the user with the values of the weights assigned to the performed actions. Then, the system compares the user profile with other user profiles, but the probability to successfully find a matching profile is low (the probability that another user has interacted with the same two services out of several thousands is low). However, a content-based approach might be used to increase such probability, by discovering other services similar to the ones the user interacted with. Then it would be possible to assess a value representing the relation between the user and the services identified as similar, even if the user has never seen them. That value might be inserted in the corresponding cell of the matrix, reducing its sparsity and increasing the probability of finding users with an overlapping profile.

Especially in SOA4All, we are going to implement this content-based approach by exploiting the semantic annotations available for services.

5. At this moment, Service-Finder includes about 27.000 services

4.3. Action Weights Decrease with Time

If a user performs exactly the same action on a specific service twice then, during the estimation of the strength of the relation between the user and the service, the current algorithm simply sums the effects of the action twice. Each time the algorithm uses the same weight assigned to that action (see line 8 of the algorithm in Figure 3), that is two executions of the same action influence equally the user-service relation: an action performed one year ago has the same weight of the same action performed five minutes ago. However, recommendations should be influenced more by recent actions rather than the old ones. For this reason, it would be better to decrease the value of the weight of an action based on the distance between the current time and the instant when the action occurred. This can be done adjusting the algorithm in Figure 3 and the way the Not-Normalized User-Service Matrix is used.

4.4. Item Popularity Influences User Similarity

The similarity between two users is calculated by taking into account the set of items rated by both users. Every item included in the set increases the similarity value: if the two users assigned the same rating to the item then the similarity is incremented a lot, while if the two users assigned opposite ratings then the increment of similarity is low or zero.

Even if this measure seems reasonable, it does not take into account the total number of users that rated a specific service. If a service has been rated by two users solely, then those two users are very related. On the contrary, if a service has been rated by almost all users, then that fact is not very significant to compute user similarities.

For this reason, the computation of user similarities might be improved by weighting the contribution that each item brings on the similarity between two users.

Acknowledgment

This research has been partially supported by the EU co-funded projects named *Service-Finder* (FP7-IST-215876)

and *SOA4All* (FP7-215219).

References

[1] Gediminas Adomavicius and Er Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17:734–749, 2005.

[2] Robert M. Bell and Yehuda Koren. Lessons from the Netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, 2007.

[3] John S. Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. pages 43–52. Morgan Kaufmann, 1998.

[4] Cyril Cleverdon. The Cranfield tests on index language devices. pages 47–59, 1997.

[5] Emanuele Della Valle, Dario Cerizza, Irene Celino, Andrea Turati, Holger Lausen, Nathalie Steinmetz, Michael Erdmann, and Adam Funk. Service-Finder: realizing Web Service Discovery at Web Scale. In *European Semantic Technology Conference (ESTC 2008)*, September, 2008.

[6] Mukund Deshpande and George Karypis. Item-Based Top-N Recommendation Algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[7] Norbert Fuhr, Th Darmstadt, and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9:223–248, 1991.

[8] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[9] Nicolas Jones and Pearl Pu. User Technology Adoption Issues in Recommender Systems. In *Networking and Electronic Commerce Research Conference*, pages 379–394, Riva del Garda, Italy, 2007.

[10] David D. Lewis and Karen Spärck Jones. Natural Language Processing for Information Retrieval. *Commun. ACM*, 39(1):92–101, 1996.

[11] Greg Linden, Brent Smith, and Jeremy York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[12] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266, New York, NY, USA, 2003. ACM.

[13] Sepideh Miralaei and Ali A. Ghorbani. Category-based Similarity Algorithm for Semantic Similarity in Multi-agent Information Sharing Systems. In *IAT '05: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 242–245, Washington, DC, USA, 2005. IEEE Computer Society.

[14] Raymond J. Mooney, Paul N. Bennett, and Loriene Roy. Book recommending using text categorization with extracted information. In *In Recommender Systems. Papers from 1998 Workshop*, pages 49–54. AAAI Press, 1998.

[15] Jacob Palme. Information Filtering. In *International Telecommunications Symposium (ITS'98)*, 9-13 Aug 1998.

[16] Dmitry Y. Pavlov and David M. Pennock. A Maximum Entropy Approach to Collaborative Filtering in Dynamic, Sparse, High-Dimensional Domains. In *In Proceedings of Neural Information Processing Systems*, pages 1441–1448. MIT Press, 2002.

[17] Michael Pazzani and Daniel Billsus. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Mach. Learn.*, 27(3):313–331, 1997.

[18] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. pages 175–186. ACM Press, 1994.

[19] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, pages 513–523, 1988.

[20] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-Based Collaborative Filtering Recommendation Algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[21] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of Recommendation Algorithms for E-Commerce. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167, New York, NY, USA, 2000. ACM.

[22] J. Ben Schafer, Joseph Konstan, and John Riedi. Recommender systems in e-commerce. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166, New York, NY, USA, 1999. ACM.

[23] Guy Shani, David Heckerman, and Ronen I. Brafman. An MDP-Based Recommender System. *J. Mach. Learn. Res.*, 6:1265–1295, 2005.

[24] Upendra Shardanand and Patti Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[25] Kirsten Swearingen and Rashmi Sinha. Interaction design for recommender systems. In *In Designing Interactive Systems 2002. ACM*, 2002.

[26] L. Ungar and D. Foster. Clustering Methods For Collaborative Filtering. In *Proceedings of the Workshop on Recommendation Systems. AAAI Press, Menlo Park California.*, 1998.

Using Context Similarity for Service Recommendation

Liwei Liu, Freddy Lecue, Nikolay Mehandjiev, Ling Xu
Centre for Service Research
The University of Manchester
Manchester, UK

***Abstract*—**Recommender systems have been successfully used to address the problem of information overload, where consumers of goods and services have too many choices and overwhelming amount of information about each choice. Here we focus on service recommendation and demonstrate the need for using multiple criteria regarding service qualities, and the need to consider multiple contextual dimensions regarding the expected use of that service. Existing service recommenders, however, fail to consider both multiple criteria and multiple context dimensions. A further problem arises since we need a reliable scalar measure for context similarity when dealing with categorical context dimensions. This need underpins the main contribution of this paper – demonstrating that concept abduction provides such a reliable measure for context similarity when the categories of a context dimension are defined as concepts in an ontology. We position this contribution within a proposed multi-context and multi-criteria approach for service recommendation based on collaborative filtering. Using experiments over a real-world dataset, we demonstrate how the concept abduction-based context similarity measure can be used to address the sparsity of data within a single context segment by allowing us to use rankings from context segments nearby.

Keywords- context similarity, concept abduction, recommender system; multicriteria recommender system; multidimensional recommender system

I. INTRODUCTION

Recommender systems (RS) have been an effective solution to *information overload* problem, under which customers can't find what they want because of the sheer volume of available information and number of alternatives. 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*" [1]. They are routinely used by e-commerce websites to help consumers make a purchasing decision.

With the development of recommender systems, the recommended objects range widely, from books, movies, music to TV programs, web pages and so on. The recommendation methods are evolved and improved in two directions: (a) covering context of use by moving from the traditional two dimensions (*user & item*) to multiple dimensions, considering context-specific dimensions of data such as use {personal or business}, and (b) covering multiple criteria by providing multiple ratings for each item.

Unexpectedly, existing work in the area of service recommenders fails to integrate both multiple context dimensions and multiple criteria, whilst both are clearly needed, because of the more personalized nature of service consumption and hence selection.

The context in which a service will be consumed plays an important role for making accurate recommendations. Adomavicius et al [4] state that a more accurate prediction of user' preference depends on the degrees of incorporation of contextual information into a recommender system [4]. This statement is based on the research of Lilien et al [5] on consumer decision making, "consumers vary in their decision-making rules because of the usage situation, the use of the good or service (for family, for gift, for self) and purchase situation (catalog sale, in-store shelf selection, salesperson aided purchase)" [5]. For instance, when recommending a vacation package, the time of the travelling, with whom the user plans to travel, stay duration, restrictions at that time and some other contextual information should be considered [2].

Dimensions of context of service use can range from scalar (ambient temperature or humidity) to categorical (using a service for work or for leisure). It is possible to provide context-aware recommendation by considering available rankings within a single context segment only, however this approach often suffers from lack of sufficient recommendations within the context segment. We can also use data from adjunct context segments, but for this we need a reliable measure of similarity (and inversely distance) between different context segments. When the context dimension is scalar, this is not a problem, yet the problem arises when we use categorical data.

Having identified the need, we propose an approach to provide such measure in a reliable and computationally efficient way. In our approach, context is modeled with a Description Logics (DLs) based ontology, to facilitate reasoning compared to other context models [6]. To calculate similarity between context segments when they are defined with categorical data using concepts from the ontology, we consider semantic similarity to evaluate the distance between these concepts. To this end we adapt the definition of Concept Abduction [7] in the context of service recommendation in

order to evaluate the rate of common description between two DL concepts.

Our method of providing reliable measure of context similarity is presented within a multi-dimensional and multi-criteria recommendation approach based on collaborative filtering (CF). It not only takes the multiple criteria and of services and also the context in which they are expected to be used into account, but also uses contextual information to reduce the sparsity problem, and use context similarity to improve the prediction. The sparsity problem is one of the classical recommendation problems [2], and we believe it will be more acute and relevant in the area of service selection, because the personalized nature of services means less feedback for a larger number of services. We demonstrate the utility of using ontology context information for better predictions, and for handling the sparsity problem by two rounds of experiments using real data on hotel service rankings under multiple contexts.

The remainder of the paper is organized as follows: we begin with introducing the relevant semantic reasoning and recommendation background in Section II. Then in Section III, the proposed approach is presented. Two rounds of experiments are implemented to evaluate the approach in Section IV. Related work is discussed in Section V, and finally we end with concluding the current work and discussing the future work.

II. BACKGROUND AND RELATED WORK

A. Semantic Similarity

In this work, we will consider semantic similarity to evaluate distance between contexts and more specially concepts, all described in a DLs- based ontology. To this end we adapt the definition of Concept Abduction [7] (Definition 1)in the context of service recommendation in order to evaluate the rate of common description between two DL concepts. In this work, we focus on comparing any concept defined in *ALN* DL.

Definition 1. (Concept Abduction)

Let L be a DL, C , D be two concepts in L , and O be a set of axioms in L . A Concept Abduction Problem (CAP), denoted as $\langle L,C,D,O\rangle$ is finding a concept $H \in L$ such that $O \models C \sqcap H \sqsubseteq D$.

This produces a compact representation of the "*difference*" i.e., Extra Description H between descriptions C and D . In other words H refers to information required by C to be D and then obtain an exact semantic similarity between C and D . The Common Description of C and D , defining as their Least Common Subsumer lcs [8], refers to information shared by C and D .

Example 1. (Extra & Common Description)

Let Fig. 1 be a sample of an *ALN* DL (E-Tourism) terminology O According to Definition 1 and the latter Figure, the Extra Description H required by concept "Man" to be (semantically) similar to concept "Traveller" is formalized as the Concept Abduction Problem: $\langle \text{L, Person, Traveller,O} \rangle$ i.e., $\exists IsTravelling.Top$. In the other hand the Common Description defined by the Least Common Subsumer of concepts "Traveller" and "Person" is referred by the information shared by both concepts, i.e., lcs(Traveler,Person)i.e., Person.

$$
\begin{aligned}
&Traveller \equiv Person \sqcap \exists IsTravelling.\top \\
&Friend \equiv Person \sqcap \exists IsAcquainted.\top \sqcap \exists IsTrusted.\top \\
&SoloTraveller \equiv Traveller \sqcap < 1.hasAccompany \\
&WithPartner \equiv Traveller \sqcap \exists IsPartner.Traveller \\
&WithOthers \equiv Traveller \sqcap \\
&\qquad \geq 1.hasTourShip \sqcap \\
&\qquad \forall hasAccompany.(\neg Friend) \sqcap \\
&\qquad \forall hasAccompany.(\neg Partner) \sqcap \\
&\qquad \forall hasAccompany.(\neg Parent) \sqcap \\
&Man \sqsubset \top, Parent \sqsubset \top, Person \sqsubset \top
\end{aligned}
$$

Figure 1. Sample of an ALN Terminology T

B. Recommender Systems

Traditionally, recommender systems are based on a single criterion, which is usually a numerical rating that presents user's preference of the whole item. Two types of entities, *users* and *items* are used for the recommendation, this giving it its two classical dimensions. This is presented as $R : Users \times Items \rightarrow Ratings$

The system is initialized by user's ratings that are either explicitly or implicitly collected. Then it tries to estimate the utility function R of the item based on these two dimensions, user and items:

$$\forall u \in Users, i_u' = \arg\max_{i \in Items} R(u,i) \qquad (1)$$

Only the items which can maximize the utility will be chosen [4]. For example, in the traditional way of recommending movies, the users are asked to rate the movies they have watched before. The recommender system predicts unknown ratings and recommends those movies with highest ratings. The recommendation approaches are usually classified into three categories: collaborative filtering, content-based and hybrid approaches [9], [4], [2], [10], [11]. Burke extends the classification and adds another three methods: demographic, utility-based and knowledge-based approaches [1]. Among them, the knowledge-based approach is quite popular [12].

C. Collaborative Filtering Approach

Among these approaches, the content-based and the *Collaborative Filtering (CF)* approaches are the two most famous recommendation approaches. The content-based approach recommends items similar to those the user liked in the past, based on the item's characteristics such as content for books, descriptions for consumer goods, etc., while the CF approach recommends to users the items liked by other users,

identified to be similar, say because of similar past rankings [10]. In the CF approach, users are asked to provide ratings as their feedback. And the historical feedback is used to find other users who have provided similar feedback on the assumption that users who had common interests in the past, tend to have similar tastes in the future [13]. GoupLens, Ringo, Amazon.com et al are all successful CF approach for prediction [14, 15]. According to [16], CF can be grouped into two classes: memory-based and model-based. For the memory-based algorithm, which is also used in this paper, prediction is computed by aggregating the ratings of other users for the same item, such as:

$$r_{u,i} = \bar{r}_u + k \sum_{u' \in U} sim(u,u') \times (r_{u',i} - \bar{r}_{u'}) \quad (2)$$

Where U denotes the set of user u and his similar u' who have rated the same item i, $r_{u,i}$ presents the rating of user u on item i, $\bar{r}_u$ is defined as the average ratings of user u, and $k = 1/\sum_{u' \in U} |sim(u,u')|$ [2].

Another key issue in the CF approach is how to locate the 'neighbors' of the active user, those users who have similar taste to the active user in the past. Pearson correlation and cosine-based approach are the two mostly used. The value of any of the two approaches ranges from -1 to +1. The greater value it is, the more similar these two users are. Thus, -1 means that the two users have exact opposite taste, and +1 means they have exact the same taste. The value is presented as $sim(u,u')$ in (2).

As CF approach is based on the customer dimension, thus it can work very well on complex items, such as music and movies. But it suffers when new items and new users appear (the New Item problem and the New User problem) and needs a large amount of initializing data before producing valuable results. Above all these, the data it gets is quite small comparing with what it needs to predict, so *sparsity of data* also becomes an acute problem [2].

D. Need for Multi-Dimensional Recommender Systems

Recommender systems are usually classified by the recommendation approaches and techniques they use, yet here we will consider the number of context dimensions and the number of criteria (ratings) they use. Most of the existing recommender systems on the market focus on a single criterion (user ranking or purchasing decision), and on two dimensions only (user and item). We call these *traditional* RS. Within the recommender systems research community, a significant volume of further work has been done, producing *extended recommender systems*, covering either multiple criteria or context dependent recommendations. However, to the best of our knowledge, the area of recommender systems which cover both in the area of service recommendation is not developed yet. However, we expect that it will be a trend to be developed because of the accuracy and personalization requirements for current service recommender systems.

E. Reduction-based Approach

One of the approaches than handle multiple contextual dimensions in recommender systems is called reduction-based approach, proposed by Adomavicius et al [4]. They argue that many traditional recommendation methods cannot be directly extend to multidimensional ones. A *reduction-based* approach can be used to estimate multidimensional ratings. It reduces multidimensional recommendations problem to the traditional two-dimensional problem. For example, in a three dimensional space $S = User \times Content \times Time$, the prediction function can be expressed by a two-dimensional one as follows [4]:

$$\forall (u,c,t) \in U \times C \times T,$$
$$\mathrm{R}^{\mathrm{D}}_{\mathrm{User} \times \mathrm{Cotent} \times \mathrm{Time}}(u,c,t) = R^{D[Time=t](User,Content,Ratings)}_{User \times Content}(u,c) \quad (3)$$

Thus the three dimensions are converted into traditional two dimensions on a particular time.

Given the sparsity problem in two-dimensional recommender system, the reduction-based approach will make the sparsity problem worse, since it groups data according to particular contextual segment. Adomavicius et al [4] use multi-level aggregation and combining with CF to reduce sparsity problem. Here, in this paper, we propose an approach based on using the ratings under similar context to predict for the active context dimensions, which at the same time will reduce sparsity problem. We assume that the more similar two contextual dimensions are, the more accurate the prediction will be.

III. PROPOSED APPROACH

As stated before, our approach not only covers multiple criteria of items but also various contextual dimensions. Since multiple criteria ratings expressed user's opinions from different aspects, which are accurate than one single total rating of the item, while contextual information describes their usage situation. Let's start with contextual description.

A. Context Concepts

Researchers hold different definitions about context. Schilit et al [27] claim that context has three important aspects, where you are, who you are with and what resources are nearby. Then they entail context including 'lighting, noise level, network connectivity, communication costs, communication bandwidth, and even the social situation' [27]. Lieberman and Selker [28] interpret context for computer systems as 'everything affects the computation except the explicit input and output', including the state of user, physical environment, computational environment, and history of user-computer environment interaction. The history here is about what the user has done, and how it will affect the future [28]. One of the most referenced context definitions is defined by Dey and Abowd [29].. They define context as "any information that can be used to characterize the situation of an entity", and the entity he defines as "a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [29].

B. Similarity of Context

Similar to the assumption of CF approach, we calculate the context similarity to show how relevant the other context dimension is to the current one. Chen states that for different context types, various quantifiable measures of the similarity between two context values are needed [24]. In this paper, we classify context into three types, *scale*, *ordinal* and *categorical*.

For scale and ordinal context types, the closer context values are, the more similar these two context dimensions are. For example, under the contextual dimension *temperature* of 25 degrees Celsius, the ratings under 20 degrees are more similar than the ratings under 35 degrees Celsius. The inverse of the numerical distance between two context values is used to compare which "candidate" context is more similar to the "active" context (we use "active" to refer to the context which our target user is considering for the use of the service we are currently recommending). Then use the weighted average formula below to predict the ratings under active context by using its related context dimensions.

$$r_p = \frac{\sum r_c / d_c}{\sum 1/d_c} \tag{4}$$

r_p is the predicted ratings under active context, while d_c is the distance between the active context and the candidate context. c presents the c th candidate context. In our example, the active context will be the 25 degrees, and the similarity between 20 degrees context and the active context is 0.2 (the inverse of context distance), and the similarity between 35 degrees context and the active context is 0.1. The similarity is better with a bigger value.

For the third type of context, i.e., categorical, we model the context types in a DL ontology. The similarity will be computed by measuring the semantic similarity from an active context C to a candidate context D using their Common Description rate. Such a rate is defined as following:

$$\mathrm{q_{cd}(C,D)} := \frac{|lcs(C, D)|}{|H| + |lcs(C, D)|} \tag{5}$$

wherein the Extra Description H is a solution of the Concept Abduction problem $\langle L, C, D, O \rangle$ (Definition 1): the difference between semantic description of active and candidate contexts C and D. In other words q_{cd} estimates the rate of descriptions which is shared by active and candidate contexts in the ontology, the higher the better is the similarity. In more details |.| refers to the size of *ALN* concept descriptions [30] i.e., |T|, |⊥|, |A|, |¬A|, |∃r| are 1; |C⊓ D| := |C|+|D|; |∀r.C| is 1 + |C|; |($\geq nr$)| and |($\leq nr$)| are 2+log(n+1) (binary encoding of n). For instance |WithOthers| is 3+(2+log(2))+ 6+2+2 i.e. 15+log(2) in Fig. 1.

Once the semantic similarity is measured, the similarity of categorical context types converts to numerical data, formula (5) can be used for predicting ratings of active categorical context.

Example 4. (Quality of Semantic Links)

Suppose concepts "SoloTraveller" and "Person" in Fig. 1, the common description rate of the latter concepts i.e., $\mathrm{q_{cd}(SoloTraveller,Person)}$ is defined by:

$$\frac{|lcs(\mathrm{SoloTraveller}, Person)|}{|H| + |lcs(SoloTraveller, Person)|} \tag{6}$$

wherein $|\mathrm{lcs(SoloTraveller,Person)}|$ is $|\mathrm{Person}| = 1$. $|\mathrm{H}| = 5 + \log(2)$. So the common description rate of these concepts $\mathrm{q_{cd}(SoloTraveller,Person)}$ is defined by: 0.18.

C. Proposed Approach

In this paper, we also assume that context has effect on users' ratings. Our approach extended CF approach with multiple criteria of ratings, while keeping in mind of reducing the side effect of sparsity problem. The approach comes following the steps below:

Firstly, determine which context dimensions have significant effect on the total rating. From the direct impression of an observer, many contextual dimensions may seem to have an effect on user's ratings, but this needs to be tested through a statistical method. If the context dimension is Scale or Ordinal, then t-test should be used. If the context dimension is Categorical, then chi-square test tends to be used. Only the contextual dimensions which have significant effect on total ratings will be taken into consideration in the further computation.

Secondly, for contextual information, select a specific context for the computation, denoted by T_m , m for the number of the specific context. For example, context of Date, it can be aggregated into four seasons, Spring, Summer, Autumn and Winter, $m = 4$. We do recommendation under Summer this specific context. The number of this kind of specific context is the combination of those context dimensions, which have significant affect on ratings. We assume that a particular hierarchy tree for each context dimension exists (if it can be hierarchy structured).

If the data under certain contextual dimension are very sparse, then the data of its similar context dimensions will be introduced. The similarity between two context dimensions is calculated through the ways expressed in Section C.

Thirdly, for multi-criteria dimensions, Euclidean distance is used to compute the distance between any two users who have rated the same item. The inverse of the distance is used as the similarity between these two users, such as mentioned in [21]. The distance between user u and u' on item i is:

$$d_i(u, u') = \sqrt{\sum_{j=0}^{n} (r_j - r'_j)^2} \tag{7}$$

n is the total number of the criteria, while r_j is the rating of user u on criterion j. Then the distance between user u and u' can be denoted as:

$$d(u,u') = \frac{1}{I}\sum_{i \in I} d_i(u,u') \tag{8}$$

I is the number of items that are rated by both user u and u'.

The similarity is denoted as the inverse of the distance:

$$sim(u,u') = \frac{1}{1 + d(u,u')} \tag{9}$$

Fourthly, predict the rankings which user u would give to the item i based on aggregating his similar users' ratings on the other items by using extended collaborative filtering prediction function in (2).

$$r_{u,i} = \overline{r}_u + k \sum_{u' \in U} sim(u,u') \times r_{u'p} \tag{10}$$

And

$$r_{u'p} = \frac{\sum (r_{u'i} - \overline{r}_{u'}) / d_c}{\sum 1/d_c}$$

$r_{u'p}$ presents the predicted rating of active context by using the candidate contexts, and $1/d_c$ is the similarity between the active context c and the candidate context. $\sum 1/d_c$ is the sum of those contexts which are used to do the prediction. And it is computed by using (4) or (5) according to its context type. In the other way, the similarities of those contexts which are not used for the prediction, will not be included in the sum.

Finally, generate recommendations on the principle that the higher overall rating, the higher chance for this service to be selected. Once the context of use is determined (automatically or through user input), since all users' ratings are stored with their context. Thus the system will match this current context with the historically computed ones, and use the data stored under that context. Then, predict the overall ratings of the items. After that, the top N items from highest score are recommended to the user.

IV. EXPERIMENTS

Two rounds of experiments are reported. The first round tests whether the prediction is better when "borrowing" ratings from a similar context dimension, compared with "borrowing" ratings from a dissimilar dimension. And the second round of experiments tests whether calculating context similarity and using it to incorporate data from all context segments with an appropriate weight (further segments still get included but with a very small weight) will produce better prediction than when not considering the weights.

In our experiments, context similarity is computed by semantic similarity functions. All the experiments were implemented using MatLab 2007b. All the experiments were run based on a XP Professional based PC with Intel Pentium(R) 4, CPU 3.20GHz, and a 1 GB of RAM.

A. Dataset

A feedback and use ratings data for an example service domain (hotel stays) is collected from tripadvisor.com. The data is suitable since a large number of feedback rankings are available from a variety of sources, and our particular set had ratings of five individual criteria, one total rating and information about the context of use. The five criteria are from Value, Rooms, Location, Cleanliness and Service. All the individual criteria ratings and the total ratings ranged from 1 to 5, with 5 as the excellent.

Apart from criteria dimensions, there are also three context dimensions, DateofStay, VisitWasFor, and TraveledWith. Two values for VisitWasFor, leisure and business, while there are eight values for TraveledWith: "with spouse", "with friends", etc. Since the raw data is too sparse to compute, we aggregate ratings of all the hotels from the same brand with the same number of stars, using the assumed equality between the levels of service and the similarity of furnishing standards of such hotels. After aggregating, there are 250,899 records, with 51,855 hotels and 109,296 users.

B. Contextual Structure

As mentioned above, when considering context, it is important to test which context dimension really affects rating estimations significantly. All the context dimensions in our dataset are categorical, so chi-square tests were applied for each context dimension and total ratings. After testing, on our data set, VisitWasFor and TraveledWith context dimensions showed the significant difference in estimating total ratings. Thus in the rest experiments, VisitWasFor and TraveledWith were considered as our context dimensions.

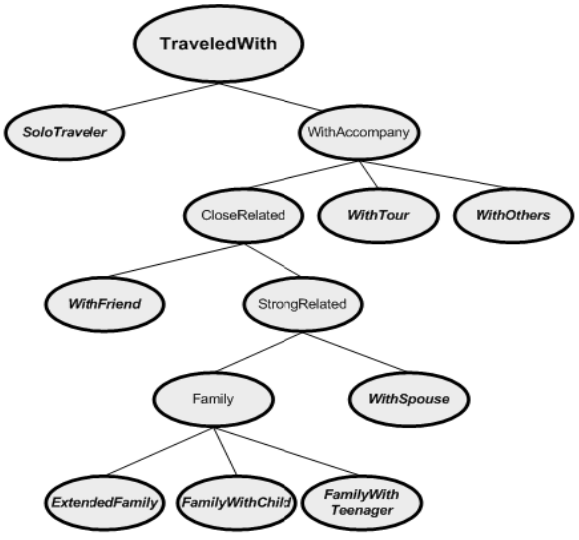

Figure 2. TraveledWith context structure

Then we used specialization taxonomy to build the structure of the two context dimensions. Fig. 2 shows the hierarchical specialization structure of TraveledWith context.

The eight circles with Bold and Italic characters are TraveledWith context values.

C. Experiment Design

The raw records were cleaned for further computation, as there were 74,066 users who only rated one hotel of the total 109,296 users. In the following experiments, users who had rated more than 3 hotels were considered. In order to get a less sparse user-hotel matrix, for the rest each record, which is defined by a user and a hotel, we calculated sum of the number of ratings for this user, and the number of ratings for this hotel. The sum was denoted as "*survival score*" of that record for the further selection. We experimented with different "survival score" thresholds to balance the number of remaining records whilst achieving a tolerable level of sparsity.

1) Effect of Similar Context Prediction: In the first round of experiments we tested whether data from similar context can be "borrowed" to predict ratings from an "active" context with insufficient number of ratings, and more importantly, if these predictions are more accurate when compared to "borrowing" data from dissimilar context.

In our dataset the VisitWasFor value of "for Business" only had records covering the TravelledWith segments of "Others" and "Solo", whilst the VisitWasFor value of Leisure had data under the TraveledWith segments of Others, Tour, Friend, Spouse, ExtendedFamily, and Solo. Therefore we used the data under "VisitWasFor" value of "Leisure" to conduct our further tests. In the dataset, the number of records under WithSpouse (4,567 records) is far more than the number of other contexts. Then WithFriend (979 records) and WithOthers (219 records) follow. Solo (131 records) and ExtendedFamily (83 records) have a few records.

According to the features of our dataset and with limited number of data, in these experiments, we choose the data under ExtendedFamily as our test set. From the TraveledWith hierarchy tree above, we can see that ExtendedFamily is close to WithSpouse, but far from WithOthers, and Solo. For the purpose of testing if the distance between context concepts on the tree affect the prediction, WithFriend dataset was not considered because it is in the middle, and it may make the results obscure. Thus we choose WithSpouse as one of the training set, WithOthers and Solo as another training set. Then we compare the recommendation prediction by using different training sets to show which context predict more accurate.

Since the records in Spouse training set (4,567 records) are much more than those in Solo and Others training set (350 records), thus to keep the experiments reasonable, only 350 records should be left in Spouse training set.

2) Effect of Prediction considering Context Similarity. This round of experiment will focus on testing the whether the cross context prediction is better with the consideration of weight. The weight is computed by the way stated in the proposed approach section.

Following the basic experiment setting in the first round of experiments, only TraveledWith context is considered, which is a categorical context type, and as structured in Fig. 2. And data under ExtendedFamily are used for testing. The semantic similarity between the five other contexts and ExtendedFamily was computed separately according to (5). Normalized context similarity is used as the weight of this context in the prediction process. For example, the similarity between Solo context and ExtendedFamily is 0.25, while the similarity between Others and ExtendedFamily is 0.5. If one record in the test set can only be predicted by one of the training sets, Solo or Others, then it is predicted based on the only one training set, without context weight consideration. However, if it can be predicted by both Solo and Others dataset, then the prediction needs to combine both predictions under the two training datasets by multiplying context weight. The context weight is the normalized context similarity, and the denominator is the sum of the similarities of the two contexts, Solo and Others, both of which can predict this record. Thus, the weight of Solo context is 0.33, and the weight of Others is 0.67.

3) Evaluation Metric: Mean Absolute Error (MAE) is used as the evaluation metric in this paper. MAE is a measure of the average absolute deviation between a predicted rating and the user's true rating.

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \tag{11}$$

N is the number of pairs of real ratings and predictions $< p_i, q_i >$. The lower MAE, the more accurate predictions are [31].

D. Experiments Results

1) Impact of Closer Context. We ran our experiments on the assumption that closer context can provide better prediction. The proposed method to used to predict the score for each user on the hotel(s) in the test set using different training sets. Then MAE was computed for these data sets.

Figure 3. Closer Context Experiment Results

As shown in Fig. 3 above, we can see that when both the numbers of the two training sets, MAE of Spouse training set is smaller, with a value of 0.8354, which means Spouse training set provides a better prediction for the test set. From Fig. 2, the contextual hierarchy tree, we can see that Spouse is closer to ExtendedFamily, than Solo and Others. The experiments support our assumption that the closer the two context dimensions are, the better prediction is. The MAEs are not very small due to the limited size of the training dataset. Then we extended the prediction on Spouse training set from 350 records to its original size, with 4567 records, a smaller MAE is obtained, which is reasonable in a very sparse level. 0.9896

in for the original spouse dataset. And sparsity level is defined as $1 - totalrecords / userNum * hotelNum$.

2) Impact of Context Weight. This experiments aim to see whether the consideration of context weight provide a better result in cross contexts prediciton.

TABLE I. CONTEXT WEIGHT EXPERIMENT RESULTS

Dataset	No. Of Records	Correlation	MAE
Whole	6192	0.3012	0.7912
WholeWeighed	6192	0.3316	0.7908

Experiments are first applied on whole dataset, which is quite sparse, with a sparsity level of 0.9877. The prediction on the whole dataset with the consideration of categorical context similarity is 0.7908, which is smaller than the prediction. without context weight is 0.7912. And the correlation between predicted value and test value in considering context weight is also better than it in the prediction without involving context weight. The results show that the consideration of context weight can provide a better prediction for cross contexts. We believe that in a larger dataset, with less sparsity level, the accuracy will be more obvious. Currently we are collecting more data, and this assumption will be tested in future work.

V. RELATED WORK

There is some work in extended recommender systems. In multidimensional recommender systems, the recommendation space is extended from traditional two dimensions $S = Users \times Items$ to multiple dimensions $S = D_1 \times D_2 \times \cdots \times D_n$. The ratings become a set of all possible ratings values, and then the rating function R is $R: D_1 \times D_2 \times \cdots \times D_n \rightarrow Ratings$, and under this rating function, RS is also called multidimensional RS [4]. In the example of recommending movies, contextual information such as time, place, and accompany persons will be considered. Because the movie the user wants to see with their partner or friend in a theater at the weekend will be different from the one watched at home with his/her parents on a weekday. Then the recommendation space will from the original two dimensions to become $S = User \times Movies \times Time \times Place \times Accompany$ [4]. Recommendation Query Language (RQL) [17] and *reduction-based* approach [4] are two approaches used for estimating the unknown ratings in a multidimensional space. Anand and Mobasher [18] define a recommendation process using context-based retrieval cues for the preference information inspired by human memory model in psychology.

One of the earliest papers on multi-criteria recommender system area is DIVA, which is initialized by a collaborative filtering database. For recommendation, it computes the distance between active user and the cases in the database and provides a ranking for users [19]. Lee and Teng [20] use collaborative filtering approach to compute the similarity of each of them separately. Subsequently, they utilized skyline queries method to identify a few good items among numerous candidates. Adomavicius and Kwon [21] believe new techniques for recommendation are needed to take full advantages of multi-criteria ratings, and they describe two approaches: a similarity-based approach and an aggregation-function-based approach. Some researchers use UTASTAR algorithm to incorporate MCDA techniques to recommendation process to provide a ranking for recommendation [22],[23]. Manouselis and Costopoulou describe and classify multi-criteria recommender systems on the basis of analyzing MCDA methods and recommendation process [3].

Up to the best of our knowledge, there are two papers that are similar to our work. Chen presented a context-aware collaborative filtering recommender system to predict a user's preferences in different context situations. He defined a method to calculate the similarity of context types which has one of three properties, categorical, continuous and hierarchical. The method uses Pearson Correlation to compute the relevance of two context values, and then the similarity returned will be used as weight for recommendation prediction [24]. This method requires the ratings of users who have rated the same hotels under different context dimensions. Thus it does not suit some datasets, such as our hotel rating dataset, because it is very rare that a user stayed at the same hotel under different contextual dimensions, such as under both for Business and Leisure. Chapphannarungsri and Maneeroj developed a multi-criteria and multi-dimensional recommender system for movie selection. They use Multiple Linear Regression for the multiple context modeling [25]. They claim that they are using multiple criteria approach, but they actually only ask user to provide an overall rating and then distribute this rating into every criterion as their value. There are some papers which are using multiple elements in the presentation vectors of items and users, such as [4], [26]. But these recommendations are actually not multi-criteria recommendation. We believe that one of the major differences between multi-criteria recommendation and single criteria recommendation is on user's feedback. Thus, do they ask the user only rate the item itself or they ask user to rate multiple criteria of the item? What Chapphannarungsri and Maneeroj do is a good way to compute the weight, but not multiple criteria recommendation.

VI. CONCLUSION AND FUTURE WORK

We conclude that we can use concept abduction to provide a reliable measure of context similarity, which can be incorporated in a service recommender system, allowing us to include data from nearby context segments with an appropriate weight, thus overcoming the data sparcity problem associated with the reduction approach to context-aware recommendation systems. Concept specialization taxonomy is used to structure these context dimensions, and help us establish the distance and similarity between categorical context segments. Indeed, we need different similarity calculations for different types of context. For scale and ordinal types, we use the inverse of the normalized distance, so that the closer the values are, the higher the similarity measure and the more similar are the two contexts. For the categorical type, the specialization hierarchy tree will allow us to establish the metric, the size of the concept difference (in description logic). Then the metric is used to compute semantic context similarity, which is used as weight of context. The semantic based approach ensures accuracy of predictions.

We incorporate this mechanism in a novel service recommendation approach which considers a number of user-oriented criteria and multiple context dimensions. Euclidean Distance is used to compute the multiple criteria ratings of users on each item. The ontology-based context modeling can keep abundant raw contextual information and also makes semantic reasoning and sharing easier, which is helpful to compute context similarity in order to make a more accurate prediction.

We have used experiments to establish that the proposal is sound, and that the recommendation is more accurate using the data under a very similar context than it using data from a remote context. The second experiment validated the consideration of semantic context similarity will provide a better prediction than the prediction on cross contexts, without taking context weight into account. One weakness of this paper is only one context dimension is considered. In the future, we will work on combining different context dimensions in one prediction.

This area of multi-criteria and multidimensional recommender system research is to be developed further, and the way in which these models and processes are integrated within the recommender systems should be carefully considered within our further research plans.

REFERENCES

[1] Burke, R., *Hybrid Recommender Systems: survey and experiments.* User Modeling and User-Adapted Interaction, 2002. **Volume 12**(Number 4): p. 331-370.

[2] Adomavicius, G. and A. Tuzhilin, *Toward the next generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.* IEEE transactions on knowledge and data engineering, 2005. **17**(6).

[3] Manouselis, N. and C. Costopoulou, *Analysis and classification of Multi-Criteria Recommender Systems* World Wide Web, 2007. **10**: p. 415-441.

[4] Adomavicius, G., et al., *Incorporating contextual information in recommender systems using a multidimensional approach.* ACM Transactions on Information Systems (TOIS), 2005. **23** (1): p. 103 - 145

[5] Lilien, G.L., P. Kotler, and S.K. Moonrthy, *Marketing Models*. 1992: Prentice Hall.

[6] Strang, T. and C. Linnhoff-Popien. *A Context modeling survey*. in *1st International Workshop on Advanced Context Modelling Reasoning and Management*. 2004.

[7] Colucci, S., et al., *Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace*, in *ECRA*. 2005. p. 41- 50.

[8] Baader, F., B. Sertkaya, and A.-Y. Turhan, *Computing the least common subsumer w.r.t. a background terminology*, in *DL*. 2004.

[9] Karta, K., *An investigation on personalized Collaborative Filtering for Web Service selection*. 2005.

[10] Balabanović, M. and Y. Shoham, *Fab: content-based, collaborative recommendation*, in *Communications of the ACM*. 1997. p. 66 - 72

[11] Leimstoll, U. and H. Stormer. *collaborative recommender systems for online shops*. in *Proceedings or the 13th Americas Conference on Information Systems*. 2007.

[12] Huang, Z., W. Chung, and H. Chen, *A graph model for e-commerce Recommender Systems.* Journal of The American Society for Information Science and Techonology, 2004. **55**(3): p. 16.

[13] Anand, S.S. and B. Mobasher, *Intelligent techniques for web personalization* in *Intelligent Techniques for Web Personalization* 2005, Springer Berlin / Heidelberg. p. 1-36.

[14] Resnick, P., et al. *GroupLens: An Open Architecture for collaborative filtering of Netnews*. in *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*. 1994.

[15] Shardanand, U. and P. Maes. *Social information filtering: algorithms for automating "word of mouth"*. in *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1995.

[16] Breese, J.S., D. Heckerman, and C. Kadie. *Empirical analysis of predictive algorithms for Collaborative Filtering*. in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. 1998.

[17] Adomavicius, G. and A. Tuzhilin, *Multidimensional Recommender Systems: A Data Warehousing approach* in e*lectronic commerce*. 2001, Springer Berlin / Heidelberg. p. 180-192.

[18] Anand, S.S. and B. Mobasher, *Contextual recommendation* in *From Web to Social Web: Discovering and Deploying User and Content Profiles*. 2007, Springer Berlin / Heidelberg. p. 142-160.

[19] Nguyen, H. and P. Haddawy. *DIVA: applying Decision Theory to Collaborative Filtering*. in *Proceedings of the Conference on Artificial Intelligence for Electric Commerce*. 1999.

[20] Lee, H.-H. and W.-G. Teng, *Incorporating Multi-criteria ratings in Recommendation Systems*, in *IEEE International Conference on Information Reuse and Integration, 2007*. 2007. p. 273-278.

[21] Adomavicius, G. and Y. Kwon, *New Recommendation techniques for multicriteria rating systems*, in *IEEE Intelligent Systems*. 2007. p. 48-55.

[22] Matsatsinis, N.F., K. Lakiotaki, and P. Delia. *A system based on multiple criteria analysis for scientific paper recommendation*. in *Proceedings of the 11th Panhellenic Conference on Informatics*. 2007.

[23] Lakiotaki, K., S. Tsafarakis, and N. Matsatsinis, *UTA-Rec: A Recommender System based on multiple criteria analysis*, in *The 2nd ACM conference of Recommender Systems*. 2008. p. 219-225.

[24] Chen, A., *Context-Aware Collaborative Filtering System: Predicting the User's Preference in the Ubiquitous Computing Environment*, in *Location- and Context-Awareness*. 2005, Springer Berlin / Heidelberg. p. 244-253.

[25] Chapphannarungsri, K. and S. Maneeroj. *Combining Multiple Criteria and Multidimension for movie Recommender System*. in *Proceedings of the International Multiconference of Engineers and Computer Scientists*. 2009.

[26] Palmisano, C., A. Tuzhilin, and M. Gorgoglione, *Using context to improve predictive modeling of customers in Personalization Applications.* IEEE transactions on knowledge and data engineering, 2008. **20**(11): p. 1535-1549.

[27] Schilit, B., N. Adams, and R. Want, *Context-aware computing applications*, in *1st International Workshop on Mobile Computing Systems and Applications*. 1994. p. 85-90.

[28] Lieberman, H. and T. Selker, *Out of context: computer systems that adapt to, and learn from, context.* IBM Systems Journal, 2000. **39**: p. 617-632.

[29] Dey, A.K., *Understanding and using context* personal and Ubiquitous Computing, 2001. **5**(1): p. 4-7.

[30] Kusters, R., *Non-Standard Inferences in Description Logu ics*, in *volume 2100 of Lecture Notes in Computer Science*. 2001, Springer.

[31] Sarwar, B., et al. *Item-based collaborative filtering recommendation algorithms*. 2001. Proceedings of the 10th international conference on World Wide Web.

Combining Collaborative Filtering and Semantic Content-based Approaches to Recommend Web Services

Freddy Lécué[1]
[1]The University of Manchester
Booth Street East, Manchester, UK
{(firstname.lastname)@manchester.ac.uk}

***Abstract*—As the abundance of web services on the World Wide Web increase, designing effective approaches for web service selection and recommendation has become more and more important. In this paper we focus on an approach dynamically offering services that fit the end-users' interests. To this end, we present a hybrid approach, coupling pure and classic collaborative-filtering methods and a semantic content-based method. On the one hand the former methods are used to automatically recommend services depending on other similar users, based on profiles, preferences and historical experience. On the other hand our semantic content-based approach performs Description Logic based reasoning on semantic descriptions of services, in order to analysis semantic similarity of services. This approach further restricts the potential results and then ensuring a semantic recommendation of services. Finally we discuss its advantages and weaknesses.**

I. Introduction

The *Semantic Web* [1], where the semantic content of the information is tagged using machine-processable languages such as the *Web Ontology Language* (OWL) [2], is considered to provide many advantages over the current "formatting only" version of the World-Wide-Web. OWL is based on concepts from Description Logics (DLs) [3] and ontologies, formal conceptualization of a particular domain. This allows us to describe the semantics of web services, e.g., their functionality in terms of input and output parameters, preconditions, effects and invariants. Such descriptions can then be used for automatic reasoning about services and automating their use to accomplish goals specified by the end-users including "intelligent" tasks such as selection, discovery and recommendation.

Due to an increasing presence and adoption of web services in the *Semantic Web*, appropriate and effective approaches for discovering, selecting and recommending services are required and key to satisfy the end-users' interests [4] in a personalised way. We focus on *Service recommendation*, defined by [5] as *the process to automatically identify the usefulness of service categories in a given situation, and then proactively discover and recommend services to the end-user*. Here, *Service recommendation* is viewed as the process of *Service selection* augmented with end-user behaviour analysis to achieve relevant and accurate service suggestion. There are three main and distinct approaches for recommending services (or any other item): collaborative filtering [6], content-based [7] approaches, and their hybrid-based version.

On the one hand collaborative filtering-based recommender systems suggest to end-users services that other (similar) end-users interacted with and appreciated in the past. Therefore, the recommendation is depending on the similarity between their profiles, preferences, interest and past rankings (or ratings). Such approaches have the positive effect to increase the serendipity in the sense that they may recommend completely different services with respect to the services the end-user has already interact with in the past. However, such a behaviour could be inappropriate in some contexts, for instance, where the end-users are not inclined to interact with services achieving total different functionalities they used to interact with. In addition, collaborative filtering-based approaches have the drawback (also called *Cold Start* problem [8]) of requiring to gather and to analyze a considerable set of end-user's interactions before being able to infer the implicit similarities among users and to provide recommendations.

On the other hand, content-based approaches reduce both latter drawbacks by analyzing the content of services, actually their characteristics such as their functionalities. The recommendation is then based on this analysis which aims at inferring similarities among services. In this direction, different levels of similarity among services' descriptions have been studied i.e., from syntactic to basic semantic based methods. While syntactic-based approaches have limitations to suggest high quality of recommendations [9], most of the semantic based-approaches [10] recommend services on basic subsumption-based ordering of their functionalities. However, the latter approaches focus only on standard semantic reasoning (i.e., subsumption) to infer semantic similarity, then reducing the accuracy of the recommendation. In addition, similarity between other main parts of services' descriptions (e.g., their preconditions and effects) are partially, or even not considered.

In this work, we suggest to unify pure collaborative filtering-based techniques with a semantic content-based approach to both i) reduce the impact of the *cold start* problem and ii) improve the semantic accuracy of services' recommendations. To this end, we exploit a semantic similarity measure, introduced by [11], and already applied to address different tasks such as services selection for their composition [12] by comparing services descriptions. Contrary to the latter work, a complete specification of services description has been considered to address possible different levels of recommendations. In more

detail, the semantic similarity of services' contents are computed through a non standard DL reasoning, aiming at evaluating the common descriptions rate of their functional categories, functional (input and output) parameters and requirements (preconditions and effects), all described as DL concepts in a domain ontology. This will ensure to provide end-users with recommendations which are semantically similar to services previously used.

The rest of the paper is organized as follows. In Section II we briefly review i) semantic web services, ii) semantic matching types and iii) DL-based common description. Section III introduces the semantic similarity measure for content-based web services recommendation. Section IV presents a prototype implementation of the unification of pure collaborative filtering-based techniques with our approach to recommend services, Section V discusses related work, and Section VI draws some conclusions.

II. Background

In this section we focus on semantics of web service (we will assume without loss of generality that each service refers to a single operation) by reviewing i) their descriptions, ii) the basic semantic matching types used to compare them, and iii) a non standard DL reasoning to infer their common descriptions and differences.

A. Semantic Web Services Descriptions

The formal model required to represent semantics of a web service s is defined as a set of semantic attributes: i) its *functional category* $\mathcal{F}(s)$, ii) its *functional parameters* i.e., inputs $In(s)$, outputs $Out(s)$ and iii) its *requirements* i.e., preconditions $\mathcal{P}(s)$, effects $\mathcal{E}(s)$, all provided by a domain ontology $\mathcal{T}$ through semantic annotations.

The particular ontology $\mathcal{T}$, which is based on the DL $\mathcal{ALE}$ [3], is part from a larger pair $\langle \mathcal{T}, \mathcal{A} \rangle$. $\mathcal{T}$ and $\mathcal{A}$ refer respectively to a Terminological Box (or TBox i.e., intentional knowledge) and an Assertional Box (or ABox i.e., extensional knowledge) in DL systems. In the following, we will focus on the TBox $\mathcal{T}$, that i) is used to annotate service descriptions, and ii) supports inference on these descriptions by means of DL reasoning. Fig. 1 shows a fragment of an example TBox $\mathcal{T}$.

According to this model, semantic web services require input parameters to be processed and preconditions to be satisfied and return some output parameters with some effects. In addition a (meta) semantic description related to its functional category is attached to each service, enabling to reason on its functionality and disambiguating services with similar functional parameters. The OWL-S profile [13], WSMO capability [14] or SA-WSDL [15] can be used to describe such services.

Example 1. ***(A Semantic Web Service)***
Suppose a service s_1 with its semantic description in the TBox $\mathcal{T}$ (Fig. 1). s_1 is defined with the `NetworkEligibility` *as functional category , which, starting from a* `Phone Number`*, a French* `Postcode` *and an* `EMail` *address (as inputs), returns the* `Network Connection` *(as an output) of the desired zone. Such a service required, as a precondition, the owner of the* `Phone Number` *and the* `EMail` *address to be the same. Finally, the service joins the returned* `Network Connection` *to the* `Phone Number` *as effect.*

$$
\begin{aligned}
&AdslEligbility \equiv NetworkEligibility \sqcap \\
&\qquad \forall hasAdslElgibility.Adsl1M \sqcap \\
&\qquad \exists hasAdslElgibility.Adsl1M \\
&NetworkConnection \equiv \forall netPro.Provider \sqcap \\
&\qquad \forall netSpeed.Speed \sqcap \exists netSpeed.Speed \\
&SlowNetworkConnection \equiv NetworkConnection \sqcap \\
&\qquad \forall netSpeed.Adsl1M \sqcap \\
&\qquad \exists netSpeed.Adsl1M \\
&Speed \equiv \forall\, mBytes.NoNilSpeed \sqcap \exists\, mBytes.NoNilSpeed \\
&Adsl1M \equiv Speed \sqcap \forall\, mBytes.1M \sqcap \exists\, mBytes.1M \\
&1M \sqsubseteq NoNilSpeed,\ Postcode\text{-}FR \sqsubset Postcode\text{-}EU \\
&PhoneNumber \equiv \forall hasDigit.FranceType \sqcap \\
&\qquad \exists hasDigit.FranceType \sqcap \\
&\qquad \forall hasAttachedNC.NetworkConnection \sqcap \\
&\qquad \exists hasAttachedNC.NetworkConnection \\
&Owner \equiv \forall hasEmail.Email \sqcap \exists hasEmail.Email \sqcap \\
&\qquad \forall hasPhoneNum.PhoneNum \sqcap \\
&\qquad \exists hasPhoneNum.PhoneNum \\
&Email \sqsubset \top, PhoneNum \sqsubset \top, NetworkEligibility \sqsubset \top
\end{aligned}
$$

Figure 1. Part of an $\mathcal{ALE}$ TBox.

B. Basic Semantic Matching Types

Given the definition of semantic web service, recommendation systems may suggest services, which have been consumed and well rated by similar end-users, based on their semantic similarity e.g., in terms of their functional parameters, categories and requirements. Such semantic similarities can be judged using a matchmaking function $Sim_{\mathcal{T}}(sd_i, sd_j)$ between two semantic descriptions sd_i, sd_j (referring to any attribute of service descriptions) encoded using the same TBox $\mathcal{T}$. The matchmaking function $Sim_{\mathcal{T}}$ goes beyond the commonly used *Exact* matching type and covers the four well known matching types [16] plus the extra matching type Intersection [17]:

- **Exact** ($\equiv$) If the concepts sd_i and sd_j are equivalent concepts; formally, $\mathcal{T} \models sd_i \equiv sd_j$.
- **PlugIn** ($\sqsubseteq$) If sd_i is sub-concept of sd_j; formally, $\mathcal{T} \models sd_i \sqsubseteq sd_j$.
- **Subsume** ($\sqsupseteq$) If sd_i is super-concept of sd_j; formally, $\mathcal{T} \models sd_i \sqsupseteq sd_j$.
- **Intersection** ($\sqcap$) If the intersection of sd_i and sd_j is satisfiable; $\mathcal{T} \not\models sd_i \sqcap sd_j \sqsubseteq \bot$;
- **Disjoint** ($\bot$) Otherwise sd_i and sd_j are incompatible i.e., $\mathcal{T} \models sd_i \sqcap sd_j \sqsubseteq \bot$;

Example 2. ***(Matching Type)***
Consider the service s_1 in Example 1 as one of the top rated service of a given end-user. Consider another service s_2 (see Table I) with `SlowNetworkConnection` *as the semantic description of its output. The semantic similarity of their outputs can be valued by a Subsume*

matching type since $\mathcal{T} \models NetworkConnection \sqsupseteq SlowNetworkConnection$ *with respect to the TBox* $\mathcal{T}$ *in Fig. 1. Therefore, the service* s_2*, with a more specific output, can be recommend to this end-user.*

The function $Sim_{\mathcal{T}}$ enables finding some levels of semantic compatibilities (i.e., *Exact*, *PlugIn*, *Subsume*, *Intersection*) and incompatibilities (i.e., *Disjoint*) among any independently defined service descriptions.

C. Common and Missing Description

Computing the matching type between semantic descriptions can be completed with a more detailed information: the DL *Missing* and *Common Descriptions* [12].

On the one hand the computation of *Missing Descriptions* is done by exploiting a non-standard DL reasoning: the *difference* or subtraction operation [11] for comparing $\mathcal{ALE}$ DL-based descriptions, thus obtaining a compact representation of the metric:

(i) the *Missing Description* $sd_j \backslash sd_i$

$$sd_j \backslash sd_i \doteq \min_{\preceq_d}\{E | E \sqcap sd_i \equiv sd_j \sqcap sd_i\} \quad (1)$$

which refers, with respect to the subdescription ordering $\preceq_d$ [18], to information required by sd_i to be semantically closer to sd_j. This defines all information which is a part of the description sd_j but not a part of the description sd_i. In case $\mathcal{T} \models sd_i \sqsupseteq sd_j$, (1) refers to information which is required by sd_i to be similar sd_j. The *Missing Description* (1) is not only necessary to explain how two descriptions are different, but also why they are different and how to make them (semantically) closer and even similar. On the other hand, the DL *Common Description* of sd_i, sd_j is:

(ii) their Least Common Subsumer [19] lcs i.e.,

$$\begin{aligned} &lcs(sd_i, sd_j) \doteq \\ &\{F | sd_i \sqsubseteq F \ \wedge \ sd_j \sqsubseteq F \ \wedge \\ &\forall F' : sd_i \sqsubseteq F' \wedge sd_j \sqsubseteq F' \Rightarrow F \sqsubseteq F'\} \end{aligned} \quad (2)$$

which refers to information shared by sd_i and sd_j.

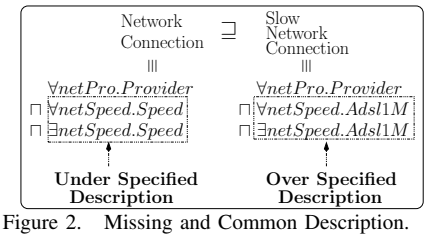

Figure 2. Missing and Common Description.

Example 3. ***(Common and Missing Description)***
The description missing in $NetworkConnection$ *to be similar to* $SlowNetworkConnection$ *is referred by* $SlowNetworkConnection \backslash NetworkConnection$ *(due to subsumption) i.e.,* $\forall netSpeed.Adsl1M \sqcap \exists netSpeed.Adsl1M$ *(Figure 2).*

The Common Description of the latter descriptions is defined by their Least Common Subsumer, which refers to the information shared by $SlowNetworkConnection$ *and the description* $NetworkConnection$ *i.e.,* $lcs(SlowNetworkConnection, NetworkConnection)$ *i.e.,* $NetworkConnection$*. In other words both descriptions are* $NetworkConnection$*.*

The DL intersection between the description $NetworkConnection$ and the Missing Description $SlowNetworkConnection \backslash NetworkConnection$ i.e., $\forall netSpeed.Adsl1M \sqcap \exists netSpeed.Adsl1M$ is of Exact matching type with `SlowNetworkConnection` i.e., perfect semantic similarity.

III. Semantic Content-based Recommendation

We consider two generic measures for evaluating semantic similarity between services descriptions: their i) *Common Description* rate, and ii) *Matching Quality*.

A. Common Description Rate

Definition 1. ***(Common Description rate)***
Given two $\mathcal{ALE}$ *semantic description* sd_i *and* sd_j*, the Common Description rate* $q_{cd} \in (0, 1]$ *provides one possible measure for the degree of similarity between* sd_i *and* sd_j*. This rate is computed using:*

$$q_{cd}(sd_i, sd_j) = \frac{|lcs(sd_i, sd_j)|}{|sd_j \backslash sd_i| \ + \ |lcs(sd_i, sd_j)|} \quad (3)$$

This rate estimates the proportion of descriptions in sd_i and sd_j which are in common. The higher the better is the similarity. The expressions in between | refer to the size of $\mathcal{ALE}$ concept descriptions ([18] p.17) i.e., $|\top|$, $|\bot|$, $|A|$, $|\neg A|$ and $|\exists r|$ is 1; $|C \sqcap D| \doteq |C| + |D|$; $|\forall r.C|$ and $|\exists r.C|$ is $1 + |C|$. For instance $|\forall netSpeed.Adsl1M \sqcap \exists netSpeed.Adsl1M|$ is 18 with respect to Fig. 1.

Example 4. ***(Common Description rate)***
Suppose services s_1 *and* s_2 *in Table I. According to* (3)*, the common description rate of the output parameters of* s_1 *and* s_2 *i.e.,* $q_{cd}(Out(s_1), Out(s_2))$ *i.e.,* $q_{cd}(NC, SlowNC)$ *is defined by:*

$$\frac{|lcs(NC, SlowNC)|}{|SlowNC \backslash NC| \ + \ |lcs(NC, SlowNC)|} \ i.e., \ \frac{2}{5} \quad (4)$$

where NC stands for NetworkConnection.

Semantic Web Service s_i		s_1	s_2
Functional Category		NetworkEligibility	AdslEligbility
Functional Parameters $\mathcal{F}(s_i)$	$In(s_i)$	Phone Number	Phone Number
		Postcode-FR	Postcode-EU
		EMail	EMail
	$Out(s_i)$	Network Connection	Slow Network Connection
Requirements	$\mathcal{P}(s_i)$	Owner	Owner
	$\mathcal{E}(s_i)$	Network Connection attached to Phone Number	Slow Network Connection attached to Phone Number

Table I
Semantic Services Descriptions.

B. Matching Quality

Definition 2. ***(Matching Quality)***
The Matching Quality q_m *between two semantic descriptions* sd_i *and* sd_j *is a value in* $(0, 1]$ *defined by* $Sim_{\mathcal{T}}(sd_i, sd_j)$ *i.e., either* 1 *(Exact),* $\frac{3}{4}$ *(PlugIn),* $\frac{1}{2}$ *(Subsume),* $\frac{1}{4}$ *(Intersection) or* 0 *(Disjoint).*

The discretization of the matching types follows a partial ordering [20] to compare and value the semantic similarity of services descriptions at matching level. Such an ordering is based on the binary and logical implication relation of Intersection from i) PlugIn and Exact and also ii) Subsume and Exact. These matching types are not all mutually exclusive, but our focus is on measuring the best achievable quality. Therefore, we assign the first matching type which is satisfied e.g., Exact rather than Plugin.

Example 5. ***(Matching Quality)***
According to the Example 2 and Definition 2, we have $q_m(Out(s_1), Out(s_2))$ *i.e.,* $q_m(NC, SlowNC)$ *is* $\frac{1}{2}$.

Contrary to q_{cd}, q_m does not estimate similarity between two descriptions but gives a more general overview and level (discretized values) of their semantic relationships by means of the subsumption relationship. We focus on a more abstract view of semantic valuation by introducing this criterion. As the common description rate, our system advertises the matching quality of different descriptions of services by pre-computing them [12].

C. A Combined Quality Model for Semantic Similarity

Given the above quality criteria, the semantic similarity of semantic descriptions sd_i and sd_j can be defined by:

$$q(sd_i, sd_j) \doteq \Big(q_{cd}(sd_i, sd_j), q_m(sd_i, sd_j) \Big) \quad (5)$$

where sd_i and sd_j can be respectively any semantic attribute of service descriptions i.e., $In(s_i)$ and $In(s_j)$; $Out(s_i)$ and $Out(s_j)$; $\mathcal{E}(s_i)$ and $\mathcal{E}(s_j)$; $\mathcal{P}(s_i)$ and $\mathcal{P}(s_j)$; $\mathcal{F}(s_i)$ and $\mathcal{F}(s_j)$ of services s_i and s_j. By considering this quality model, we aim at evaluating the level of semantic similarity between two different services descriptions.

In case some semantic attributes of services are defined by multiple semantic descriptions e.g., $In(s_1)$ and $In(s_2)$ in Table I, the value of each quality criterion is retrieved by computing their average. This average is computing independently along each dimension of the quality model.

Example 6. ***(Multiplicity in Attributes Description)***
Suppose the input parameters of s_1 *and* s_2 *in Table I. The semantic similarity of* $In(s_1)$ *and* $In(s_2)$ *is defined by the quality vector* $q(In(s_1), In(s_2))$*:*

$$\frac{1}{3} \times \Big(\sum_{k=1}^{3} q_{cd}(In^k(s_1), In^k(s_2)), \sum_{k=1}^{3} q_m(In^k(s_1), In^k(s_2)) \Big)$$
$$i.e., \quad (1, \frac{11}{12}) \quad (6)$$

$In^k(s_i)$ *refers to the* k^{th} *input parameter of* $s_{i, i \in \{1,2\}}$.

In case the number of semantic descriptions are different between attributes of services, only comparable (in term of subsumption) pairwise of descriptions are considered.

The quality (5) for semantic similarity can be generalised to any pair of services s_i and s_j rather than to any pair of semantic descriptions (or services attributes) as:

$$\overset{*}{q}(s_i, s_j) \doteq \sum_{l \in \{\mathcal{F}, In, Out, \mathcal{P}, \mathcal{E}\}} \omega_l \times q\Big(l(s_i), l(s_j) \Big) \quad (7)$$

where $\omega_l \in [0, 1]$ is the weight assigned to the l^{th} service description attribute and $\sum_{l \in \{\mathcal{F}, In, Out, \mathcal{P}, \mathcal{E}\}} \omega_l = 1$. In this way preferences on quality one some desired service attribute can be done by simply adjusting ω_l e.g., the functional category of a service could be weighted higher. Finally, the results returned by (7) is a pair of values in $[0, 1] \times [0, 1]$ referring to the common description rate and matching quality between services s_i and s_j.

Example 7. ***(Semantic Similarity of services)***
Suppose services s_1 *and* s_2 *in Table I. According to Examples 4, 5 and 6, we obtained respectively* $q_m(Out(s_1), Out(s_2))$, $q_{cd}(Out(s_1), Out(s_2))$, $q_m(In(s_1), In(s_2))$ *and* $q_{cd}(In(s_1), In(s_2))$. *The other quality of services descriptions attributes are computed using the same process along* $\mathcal{F}$, $\mathcal{P}$, *and* $\mathcal{E}$ *(see Table II). Finally, by means of* (7)*, we obtain:*

$$\overset{*}{q}(s_1, s_2) \doteq (\frac{69}{100}, \frac{47}{60}) \quad (8)$$

where the weight ω_l *is* $\frac{1}{5}$ *for each attribute.*

The quality of semantic similarity between services can be then compared by analysing $\overset{*}{q}$ i.e., their q_{cd} and q_m elements. For instance $\overset{*}{q}(s_i, s_j) > \overset{*}{q}(s_i, s_k)$ if both the common description rate and matching quality of $\overset{*}{q}(s_i, s_j)$ are higher than $\overset{*}{q}(s_i, s_k)$. Alternatively, in case of conflicts e.g., the value of the first element of $\overset{*}{q}(s_i, s_j)$ is better than the first element of $\overset{*}{q}(s_i, s_k)$ but worse for the second element, we compare a weighted average (with a weight of $\frac{1}{2}$) of their normalised components.

$q(l(s_i), l(s_j))$	$q_{cd}(l(s_i), l(s_j))$	$q_m(l(s_i), l(s_j))$
$q(\mathcal{F}(s_1), \mathcal{F}(s_2))$	$\frac{1}{20}$	$\frac{1}{2}$
$q(In(s_1), In(s_2))$	1	$\frac{11}{12}$
$q(Out(s_1), Out(s_2))$	$\frac{2}{5}$	$\frac{1}{2}$
$q(\mathcal{P}(s_1), \mathcal{P}(s_2))$	1	1
$q(\mathcal{E}(s_1), \mathcal{E}(s_2))$	1	1

Table II
QUALITY ALONG DIFFERENT ATTRIBUTES.

IV. ARCHITECTURE AND IMPLEMENTATION

Our technique of "*semantic content-based recommendation*" is implemented and integrated with state-of-the-art filtering techniques-based approaches. We describe the prototype architecture and discuss the extension of the basic collaborative filtering techniques we have suggested to deal with semantic similarity of services descriptions.

A. General Overview

The prototype architecture (Figure 3) consists of four main state-of-the-art modules, namely i) the *Collaborative Filtering* module[1] (based on Taste, an extensible framework that implements many recommendation algorithms available in literature) which is the core of the recommendation system, ii) a *Monitoring and Management* infrastructure (using Active BPEL[2]) responsible for

[1] http://taste.sourceforge.net
[2] http://sourceforge.net/projects/activebpel/

tracking logs and behaviours of end-users, iii) an *End-User Behaviour Correlation Analyser* which estimates the relations between users and services by examining the end-user rates (on services), history e.g., actions performed by end-users (through analysis of logs), and iv) a *Semantic Reasoning* module (DL reasoner Fact++ [21]) responsible for specific DL inferences such as subsumption (e.g., matching quality), difference (Common description rate).

In addition, a pool of (SA-WSDL) semantic-based services (based on the Minimal Service Model[3]) are stored in a RDF[4] repository [22]. Their descriptions are based on an $\mathcal{ALE}$ TBox (formally defined by 1100 concepts and 390 properties). The users profiles, required to evaluate similarity between end-users, and preferences (e.g., ranking about services) are described with RDF based FOAF[5].

Finally, our architecture is extended by our *Semantic Content-based* approach, responsible for computing and ranking semantic similarities between services using (7).

B. Limitation of Pure Collaborative Filtering Approaches

In a nutshell, the pure collaborative filtering-based approaches aim at producing personal recommendations of web services by computing the similarity between profiles, behaviours and preferences of different end-users. To this end, they simply requires the reference to an active end-user (i.e., end-user expecting services recommendations) and the latter personal information as inputs. Then, the personal information of active end-users' neighbours are jointly considered in order to identify the most appreciated services (using a ranking). From this set of services, the ones that the active end-user has not yet interact with are recommended. Most of recommendation systems exploits such end-users similarities in terms of their personal information to select and then recommend relevant services. However, as mentioned in Section I, such systems may recommend completely different services (in term of their functionality) with respect to the ones the active end-user has already interact with in the past.

C. Our Integrated Approach

Towards the latter issue we suggest to extend the previous approach by also recommending services based on the semantic similarities between their descriptions and the services used by similar end-users. To this end, our recommendation system requires the reference of an active end-user (and its personal information) and some services she used to interact with in the past as inputs. Firstly, our approach considers the neighbours of the active end-user by computing similarities between different end-users personal information. Then, services manipulated by similar end-users, except the services already used by the active end-user, are ranked depending on their semantic similarity (by means of the *Semantic Content-based Approach* introduced in Section III) with services the active end-user used to interact with. Finally, the top

[3]http://cms-wg.sti2.org/TR/d12/v0.1/

[4]http://www.w3.org/RDF/

[5]http://www.foaf-project.org/

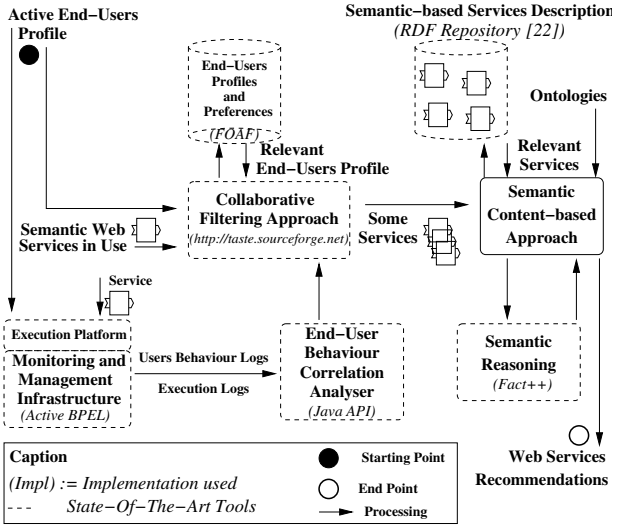

Figure 3. Core Architecture.

k services are then recommended to the active end-user. Therefore, the active end-user of our platform will receive simple and intuitive indications about potentially useful services, without having to deal with any configuration or data request. Our system analyses the end-user actions and interests, giving back the most suitable suggestions.

Even if we focused on coupling both collaborative filtering techniques and semantic similarities for recommending services, it is straightforward to adapt our approach in order to consider only one of the previous approaches.

V. Related Work

The recommendation approaches [23] are usually classified into three categories: collaborative filtering [6], content-based [7] and hybrid approaches. [24] extended this classification by introducing demographic, utility-based and knowledge-based approaches.

Among these approaches, the content-based and the collaborative filtering approaches are the two most famous recommendation approaches. The content-based approach recommends items (or services in our context) similar to those the end-users appreciates (or used to interact with) with in the past, based on the item's characteristics such as their content while the collaborative filtering (or end-user based) approaches recommend to end-users the items liked by other users, identified to be similar, say because of similar profiles, preferences, interest and past rankings (or even ratings). Our work presents a comprehensive study of how to provide accurate recommendation by systematically combining the (semantic) content-based method and a classic collaborative filtering-based method.

There is limited work in the literature which employs semantic content-based methods for web service recommendation. Indeed, there is no real semantic analysis of similar contents but rather enhanced syntactic comparison of contents [9]. However, the recommendation systems should benefit of recent results in the research area of

semantic computing such as DL reasoning [21], [18], [19] and semantic similarity measures [25].

Thus, we suggested to enhance the recommendation systems with non standard DL reasoning on services contents i.e., DL difference [11]. However, other approaches such as the i) difference operator [26] or ii) Concept Abduction [27] can be used to compute from a given description all the information different (and so similar) in another description. On the one hand (1) is a refinement of [26]'s difference that considers the syntactic minimum ($\preceq_d$) between incomparable $\mathcal{ALE}$ descriptions instead of a semantic maximum (ordering according to the subsumption operator). The result of the former does not contain redundancies and its result is more readable by a human user. On the other hand concept abduction considers $\mathcal{ALN}$ DLs. [26] and (1) perform an equivalence between two concept descriptions ($\mathcal{T} \models E \sqcap sd_i \equiv sd_j$ or $E \sqcap sd_i \equiv sd_j \sqcap sd_i$) whereas the concept abduction computes a subsumption of concept descriptions ($\mathcal{T} \models E \sqcap sd_i \sqsubseteq sd_j$).

VI. Conclusion

We introduced a semantic content-based recommendation system that provides end-users with recommendations about semantic web services that could be of their interest. To this end our approach suggests recommendations of services by combining state-of-the-art collaborative filtering approaches (based on similarity in the services' usage) and a semantic content-based approach (based on the semantic similarity of services' contents).

Even if our approach is appropriate to recommend services in an accurate way with high level of semantic descriptions, there are some issues regarding its scalability. Indeed, the DL reasoning part is the most time consuming process in our architecture. This is caused by the critical complexity of q_{cd} computation through DL Difference, LCS and subsumption (even in $\mathcal{ALE}$ DL). Indeed, deciding subsumption in $\mathcal{ALE}$ is NP-complete. There is a trade-off between semantic expressivity of services descriptions and quality/relevance of recommendation.

As future work, we expect to perform experimentations on real-world applications to consider complexity vs. usability, and to optimise DL reasoning to scale up the overall process of recommendation. We also plan comparative experiments with other discovery and matchmaking approaches [28] to evaluate services similarity. It would be also interesting to further explore more refined recommendations by considering i) specific tasks and goals the end-users expect to achieve, and ii) the current context.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[2] M. K. Smith, C. Welty, and D. L. McGuinness, "Owl web ontology language guide," W3C Recommendation, 2004.

[3] F. Baader and W. Nutt, "Basic description logics," in *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[4] T. M. Eap, M. Hatala, and D. Gasevic, "Enabling user control with personal identity management," in *IEEE SCC*, 2007, pp. 60–67.

[5] A. Moon, Y.-M. Park, and Y.-I. Choi, "Ontology based knowledge modeling for the two-step personalized services in next generation networks," in *ICSOFT (2)*, 2009, pp. 332–337.

[6] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *SIGIR*, 2007, pp. 39–46.

[7] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*, 2007, pp. 325–341.

[8] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *SIGIR*, 2002, pp. 253–260.

[9] M. B. Blake and M. F. Nowlan, "A web service recommender system using enhanced syntactical matching," in *ICWS*, 2007, pp. 575–582.

[10] H. Xia and T. Yoshida, "Web service recommendation with ontology-based similarity measure," in *ICICIC*. Washington, DC, USA: IEEE Computer Society, 2007, p. 412.

[11] S. Brandt, R. Küsters, and A.-Y. Turhan, "Approximation and difference in description logics," in *KR*, 2002, pp. 203–214.

[12] F. Lécué and A. Delteil, "Making the difference in semantic web service composition." in *AAAI*, 2007, pp. 1383–1388.

[13] A. Ankolenkar, M. Paolucci, N. Srinivasan, and K. Sycara, "The owl services coalition, owl-s 1.1 beta release," Tech. Rep., July 2004.

[14] D. Fensel, M. Kifer, J. de Bruijn, and J. Domingue, "Wsmo submission, w3c member submission," 2005.

[15] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller, "Adding semantics to web services standards," in *ISWC*, 2003, pp. 395–401.

[16] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *ISWC*, june 2002, pp. 333–347.

[17] L. Li and I. Horrocks, "A software framework for matchmaking based on semantic web technology," in *WWW*, 2003, pp. 331–339.

[18] R. Küsters, *Non-Standard Inferences in Description Logics*, ser. Lecture Notes in Computer Science. Springer, 2001.

[19] W. W. Cohen, A. Borgida, and H. Hirsh, "Computing least common subsumers in description logics," in *AAAI*, 1992, pp. 754–760.

[20] F. Lécué, O. Boissier, A. Delteil, , and A. Léger, "Web service composition as a composition of valid and robust semantic links," *IJCIS*, vol. 18, no. 1, March 2009.

[21] I. Horrocks, "Using an expressive description logic: Fact or fiction?" in *KR*, 1998, pp. 636–649.

[22] C. Pedrinaci, D. Lambert, M. Maleshkova, D. Liu, J. Domingue, and R. Krummenacher, *Adaptive Service Binding with Lightweight Semantic Web Services*, 2010, ch. Service Engineering: European Research Results.

[23] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, 2005.

[24] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[25] V. Cordì, P. Lombardi, M. Martelli, and V. Mascardi, "An ontology-based similarity between sets of concepts," in *WOA*, 2005, pp. 16–21.

[26] G. Teege, "Making the difference: A subtraction operation for description logics," in *KR*, 1994, pp. 540–550.

[27] S. Colucci, T. D. Noia, E. D. Sciascio, F. M. Donini, and M. Mongiello, "Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace," *Electronic Commerce Research and Applications*, vol. 4, no. 4, pp. 345–361, 2005.

[28] M. Klusch, P. Kapahnke, and I. Zinnikus, "Sawsdl-mx2: A machine-learning approach for integrating semantic web service matchmaking variants," in *ICWS*, 2009, pp. 335–342.

Using Contextual Information for Service Recommendation

Abstract

Recommender systems have successfully supported the effective and efficient selection of one product out of the many which meet consumer's needs. Our work extends this work to the area of service recommendation, where we demonstrate the need for using multiple criteria regarding service qualities, and the need to consider multiple contextual dimensions regarding the expected use of that service. This motivates our proposed approach, which uses collaborative filtering and considers both multiple ranking criteria and a number of different context dimensions. The expected sparsity of ranking is dealt with when handling the contextual information by introducing the metric of concept similarity for different context types, and showing how this metric can help reuse data between contexts. At the end of the paper, two rounds of experiments are described. The first shows that considering context produces better predictions, and the second round is used to test our approach for handling sparsity by reusing data between contexts using the similarity metric.

1. Introduction

The exponential growth of the World-Wide-Web and the emergence of e-commerce sites has caused *information overload* problem, under which customers can't find what they wanted in a short time and are often lost during the searching process, and also the sheer volume of available information also make it hard to judge its reliability and trustworthiness for customers. Recommender systems (RS) have been an effective solution to this problem. 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" [1]. They are routinely used by e-commerce websites to help consumers make a purchasing decision. Their use can increase e-commerce sales in three ways: by converting browsers into buyers, increasing cross-sell by suggesting additional products and building customer loyalty through "creating a value-added relationship between the site and the customer" [2].

With the development of recommender systems, the recommended objects range widely, from books, movies, music to TV programs, web pages and so on. And generally the recommendation methods also develop in two ways. In the first way, the recommendation space converts from the traditional two dimensions (*user & item*) to multiple dimensions, considering context-specific dimensions of data such as use{*personal or business*}. In the second way, the ratings change from only one rating per item to multiple criteria ratings.

In this paper, we show that both these developments are necessary for selecting services, yet existing work in the area fails to integrate both multiple context dimensions and multiple criteria. Having identified the need, we propose a recommendation method for the selection of services. Indeed, from a broad perspective, service recommendation can be considered identical to product selection, yet service recommendation needs to be more personalized, with both context and criteria considerations playing an important role. For example, when recommending a vacation package, the time of the travelling, with whom the user plans to travel, stay duration, restrictions at that time and some other contextual information should be considered [3]. In recommending a restaurant, the food and service, environment should be taken into consideration as the multiple criteria. Manouselis and Costopoulou point out that Multiple Criteria Decision Making (MCDM) methods can facilitate the recommendation process [4]. MCDM approaches are aligned with situations where the decision is high-value, and users have significant time to select criteria weights and rank products according to their criteria. This suggests that effective RS support would need innovative combination of speed and multiple criteria based selection. And our work strives to deliver better recommendations for service by taking into account both multiple criteria of services and also the context in which they are expected to be used.

Based on collaborative filtering (CF) approach, we propose a novel approach which not only takes the multiple criteria and context of use into account, but also uses contextual information to reduce the

sparsity problem. The sparsity problem is one of the classical recommendation problems [3], and we believe it will be more acute and relevant in the area of service selection, because the personalized nature of services means less feedback for a larger number of services. We demonstrate the utility of using context information for better predictions, and for handling the sparsity problem by two rounds of experiments using real data on hotel service rankings under multiple contexts.

The remainder of the paper is organized as follows: we begin with introducing the recommendation backgrounds and related work in Section 2. Then in Section 3, the proposed approach is presented. Two rounds of experiments are implemented to evaluate the approach in Section 4, and finally we conclude the current work and discuss the future work in Section 5.

2. Background and related work

2.1 Recommender systems

Recommender systems have become an important research area since mid-1990s. Traditionally, recommender systems are based on a single criterion, which is usually a numerical rating that presents user's preference of the whole item. Two types of entities, users and items are used for the recommendation, this giving it its two classical dimensions. This is presented by the following:

$$R: Users \times Items \rightarrow Ratings \quad (1)$$

The system is initialized by user's ratings that are either explicitly or implicitly collected. Then it tries to estimate the utility function R of the item based on these two dimensions, user and items:

$$\forall u \in Users, i_u' = \arg\max_{i \in Items} R(u,i) \quad (2)$$

Only the items which can maximize the utility will be chosen [5]. For example, in the traditional way of recommending movies, the users are asked to rate the movies they have watched before. The recommender system predicts unknown ratings and recommends those movies with highest ratings. The recommendation approaches are usually classified into three categories: collaborative filtering, content-based and hybrid approaches [6], [5], [3], [7], [8]. Burke extends the classification and adds another three methods: demographic, utility-based and knowledge-based approaches [1]. Among them, the knowledge-based approach is quite popular [9].

2.2 Collaborative filtering approach

Among these approaches, the content-based and the *Collaborative Filtering* (CF) approaches are the two most famous recommendation approaches. The content-based approach recommends items similar to those the user liked in the past, based on the item's characteristics such as content for books, descriptions for consumer goods, etc., while the CF approach recommends to users the items liked by other users, identified to be similar, say because of similar past rankings [7]. In the CF approach, users are asked to provide ratings as their feedback. And the historical feedback is used to find other users who have provided similar feedback on the assumption that users who had common interests in the past, tend to have similar tastes in the future [10]. GoupLens, Ringo, Amazon.com et al are all successful CF approach for prediction [11, 12]. According to [13], CF can be grouped into two classes: memory-based and model-based. For the memory-based algorithm, which is also used in this paper, prediction is computed by aggregating the ratings of other users for the same item, such as:

$$r_{u,i} = \bar{r}_u + k \sum_{u' \in U} sim(u,u') \times (r_{u',i} - \bar{r}_{u'}) \quad (3)$$

Where U denotes the set of user u and his similar u'who have rated the same item i, $r_{u,i}$ presents the rating of user u on item i, $\bar{r}_u$ is defined as the average ratings of user u, and $k = 1/\sum_{u' \in U} |sim(u,u')|$ [3].

Another key issue in the CF approach is how to locate the 'neighbors' of the active user, those users who have similar taste to the active user in the past. Pearson correlation and cosine-based approach are the two mostly used. The value of any of the two approaches ranges from -1 to +1. The greater value it is, the more similar these two users are. Thus, -1 means that the two users have exact opposite taste, and +1 means they have exact the same taste. The value is presented as $sim(u,u')$ in function (3).

As CF approach is based on the customer dimension, thus it can work very well on complex items, such as music and movies. But it suffers when new items and new users appear (the New Item problem and the New User problem) and needs a large amount of initializing data before producing valuable results. Above all these, the data it gets is quite small comparing with what it needs to predict, so sparsity of data also becomes an accute problem [3].

2.3 Classification of recommender systems

Recommender systems are usually classified by the recommendation approaches and techniques they use. We propose an alternative novel framework,

where we classify recommender systems into four types according to the number of decision criteria and number of (context) dimensions they consider as shown in Figure 1. *Criteria* in recommender system are usually presented as the users' ratings of different features (or properties) of the items. Criteria are thus about the item itself. This contrasts with *Context Dimensions*, which represent different type of context parameters relevant to the selection.

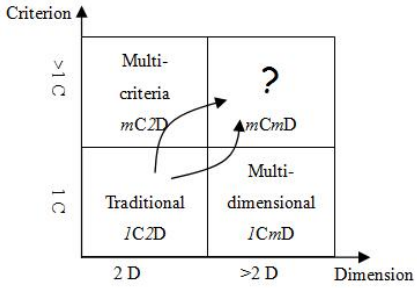

Figure 1. Recommender systems

The four types of recommender systems are *1*C*2*D (single-criterion 2-dimensional systems, or "*traditional*" systems), *1*C*m*D (single-criterion *multi-dimensional systems*), *m*C*2*D (*multi-criteria* 2-dimensional systems) and *m*C*m*D (*multi-criteria multi-dimensional*) recommender systems. Most of the existing recommender systems on the market are *1*C*2*D recommender systems, though within the recommender systems research community, a significant volume of further work has been done. In this paper, *1*C*2*D recommender systems are also referred to as traditional, while the other three are *extended systems*, considering the current applications in e-commerce. However, to the best of our knowledge, the area of *m*C*m*D recommender systems is not developed yet. We expect that it will be a trend to be developed because of the accuracy and personalization requirements for current recommender systems.

2.4 Existing Extended RS

In multidimensional recommender systems, the recommendation space is extended from traditional two dimensions $S = Users \times Items$ to multiple dimensions $S = D_1 \times D_2 \times \cdots \times D_n$. The ratings become a set of all possible ratings values, and then the rating function R is $R: D_1 \times D_2 \times \cdots \times D_n \rightarrow Ratings$, and under this rating function, RS is also called multidimensional RS [5]. Anand and Mobasher [14] define a recommendation process using context-based retrieval cues for the preference information inspired by human memory model in psychology. In the example of recommending movies, contextual information such as time, place, and accompany persons will be considered. Because the movie the user wants to see with their partner or friend in a theater at the weekend will be different from the one watched at home with his/her parents on a weekday. Then the recommendation space will become $S = User \times Movies \times Time \times Place \times Accompany$ from the original two dimensions [5]. Recommendation Query Language (RQL) [15] and reduction-based approach [5] are two approaches used for estimating the unknown ratings in a multidimensional space. Reduction-based approach reduces multidimensional recommendations problem to the traditional two-dimensional problem. For example, in a three dimensional space $S = User \times Content \times Time$, the prediction function can be expressed as the prediction in a two dimensional space $S' = User \times Content$ when $Time = t$ [5].

One of the earliest papers on multi-criteria recommender system area is DIVA, which is initialized by a collaborative filtering database. For recommendation, it computes the distance between active user and the cases in the database and provides a ranking for users [16]. Lee and Teng [17] use collaborative filtering approach to compute the similarity of each of them separately. Subsequently, they utilized skyline queries method to identify a few good items among numerous candidates. Adomavicius and Kwon [18] believe new techniques for recommendation are needed to take full advantages of multi-criteria ratings, and they describe two approaches: a similarity-based approach and an aggregation-function-based approach. Some researchers use UTASTAR algorithm to incorporate MCDA techniques to recommendation process to provide a ranking for recommendation [19],[20]. Manouselis and Costopoulou describe and classify multi-criteria recommender systems on the basis of analyzing MCDA methods and recommendation process [4].

Up to the best of our knowledge, there are two papers that are similar to our work. Chen presented a context-aware collaborative filtering recommender system to predict a user's preferences in different context situations. He defined a method to calculate the similarity of context types which has one of three properties, categorical, continuous and hierarchical. The method uses Pearson Correlation to compute the relevance of two context values, and then the similarity returned will be used as weight for recommendation prediction [21]. This method requires the ratings of users who have rated the same hotels under different context dimensions. Thus it does not suit to some dataset, such as our hotel rating dataset, because it is very rare that a user stayed at

the same hotel under different contextual dimensions. For instance, under both for Business and Leisure. Chapphannarungsri and Maneeroj developed a multi-criteria and multi-dimensional recommender system for movie selection. They use Multiple Linear Regression for the multiple context modeling [22]. They claim that they are using multiple criteria approach, but they actually only ask user to provide an overall rating and then distribute this rating into every criterion as their value. There are some papers which are using multiple elements in the presentation vectors of items and users, such as [5], [23]. But these recommendations are actually not multi-criteria recommendation. We believe that one of the major differences between multi-criteria recommendation and single criteria recommendation is on user's feedback. Thus, do they ask the user only rate the item itself or they ask user to rate multiple criteria of the item? What Chapphannarungsri and Maneeroj do is a good way to compute the weight, but not multiple criteria recommendation.

3. Recommendation based on context similarity

As stated before, our approach not only covers multiple criteria of items but also various contextual dimensions. Since multiple criteria ratings expressed user's opinions from different aspects, which are accurate than one single total rating of the item, while contextual information describes their usage situation. Let's start with contextual description.

3.1 Context concepts

Researchers hold different definitions about context. Schilit et al [24] claim that context has three important aspects, where you are, who you are with and what resources are nearby. Then they entail context including 'lighting, noise level, network connectivity, communication costs, communication bandwidth, and even the social situation' [24]. Lieberman and Selker [25] interpret context for computer systems as 'everything affects the computation except the explicit input and output', including the state of user, physical environment, computational environment, and history of user-computer environment interaction. The history here is about what the user has done, and how it will affect the future [25]. One of the most referenced context definitions is defined by Dey and Abowd [26].. They define context as "any information that can be used to characterize the situation of an entity", and the entity he defines as "a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [26].

Context plays an important role to make more accurate recommendation prediction. Adomavicius et al [5] state that a more accurate prediction of user' preference depends on the degrees of incorporation of contextual information into a recommender system [5]. This statement is based on the research of Lilien et al [27] on consumer decision making, "consumers vary in their decision-making rules because of the usage situation, the use of the good or service (for family, for gift, for self) and purchase situation (catalog sale, in-store shelf selection, salesperson aided purchase)" [27].

3.2 Similarity of Context

Similar to the assumption of CF approach, we calculate the context similarity to show how relevant the other context dimension is to the current one. Chen states that for different context types, various quantifiable measures of the similarity between two context values are needed [21]. In this paper, we classify context into three types, *scale*, *ordinal* and *categorical*.

For scale and ordinal context types, the closer context values are, the more similar these two context dimensions are. For example, under the contextual dimension *temperature* of 25 degrees Celsius, the ratings under 20 degrees are more similar than the ratings under 35 degrees Celsius. The inverse of the numerical distance between two context values is used to compare which "candidate" context is more similar to the "active" context (we use "active" to refer to the context which our target user is considering for the use of the service we are currently recommending). Then use the weighted average formula below to predict the ratings under active context by using its related context dimensions.

$$r_p = \frac{\sum r_c / d_c}{\sum 1 / d_c} \qquad (4)$$

r_p is the predicted ratings under active context, while d_c is the distance between the active context and the candidate context. c presents the c th candidate context. The bigger distance is, the less similar of these two contexts. In our example, the active context will be the 25 degrees, and the similarity between 20 degrees context and the active context is 0.2, and the similarity between 35 degrees context and the active context is 0.1. Opposite to the context distance, the context similarity is better with a bigger value.

For the third type of context, the categorical type, we can model the different context types in a

specialization taxonomy, or an ontology. This will contain a hierarchical tree of domain concepts and the way they specialize one another, and an attribute-value mechanism to describe crucial properties of each concept [28]. After establishing the hierarchical structure (assume the structure is established under the aim of finding similar contexts), the similar contextual dimensions stay in a closer level in the hierarchy tree. The similarity will be computed by measuring the distance (say by counting the number of the specialization links, or graph edges) from active context to candidate context. Then the similarity of categorical context types converts to numerical data, the formula (4) can be used for predicting ratings of active categorical context.

3.3 Proposed Approach

In this paper, we also assume that context has effect on users' ratings. Our approach extended CF approach with multiple criteria of ratings, while keeping in mind of reducing the side effect of sparsity problem. The approach comes following the steps below:

Firstly, determine which context dimensions have significant effect on the total rating. From the direct impression of an observer, many contextual dimensions may seem to have an effect on user's ratings, but this needs to be tested through a statistical method. If the context dimension is Scale or Ordinal, then t-test should be used. If the context dimension is Categorical, then chi-square test tends to be used. Only contextual dimensions which have significant effect on total ratings will be taken into consideration in the further computation.

Secondly, for contextual information, select a specific context for the computation, denoted by T_m, m for the number of the specific context. For example, context of Date, it can be aggregated into four seasons, Spring/Summer/Autumn/Winter, $m = 4$. We do recommendation under Summer this specific context. The number of this kind of specific context is the combination of those context dimensions, which have significant affect on ratings. We assume that a particular hierarchy tree for each context dimension exists (if it can be hierarchy structured).

Given the sparsity problem in two-dimensional recommender system, the reduction-based approach will make the sparsity problem worse, since it groups data according to particular contextual segment. Referring to the sparsity problem, we use the ratings under similar context to predict for the active context dimensions based on the assumption that the more similar two contextual dimensions are, the more accurate the prediction will be. The similarity between two context dimensions is calculated through the ways expressed in Section 3.2, and then will be combined in the recommendation prediction.

Thirdly, for multi-criteria dimensions, Euclidean distance is used to compute the distance between any two users who have rated the same item. The inverse of the distance is used as the similarity between these two users, such as mentioned in [18]. The distance between user u and u' on item i is:

$$d_i(u,u') = \sqrt{\sum_{j=0}^{n} (r_j - r'_j)^2} \qquad (5)$$

n is the total number of the criteria, while r_j is the rating of user u on criterion j.

Then the distance between user u and u' can be denoted as:

$$d(u,u') = \frac{1}{I}\sum_{i \in I} d_i(u,u') \qquad (6)$$

I is the number of items that are rated by both user u and u'.

The similarity is denoted as the inverse of the distance:

$$sim(u,u') = \frac{1}{1+d(u,u')} \qquad (7)$$

Fourthly, predict the rankings which user u would give to the item i based on aggregating his similar users' ratings on the other items, and those ratings under closer context will have a higher weight in prediction, by using extended collaborative filtering prediction function as follows.

$$r_{u,i} = \bar{r}_u + k\sum_{u' \in U} sim(u,u') \times r_{u'p} \qquad (8)$$

And $$r_{u'p} = \frac{\sum (r_{u'i} - \bar{r}_{u'}) / d_c}{\sum 1/d_c}$$

$r_{u'p}$ presents the predicted rating of active context by using the candidate contexts, d_c is the distance between the active context and the candidate context, as expressed in Section 3.2. Thus the bigger the distance of between active and candidate contexts is, the less consideration when using that candidate context to predict for active context.

Finally, generate recommendations on the principle that the higher overall rating, the higher chance for this service to be selected. Once the context of use is determined (automatically or through user input), since all users' ratings are stored with their context. Thus the system will match this current context with the historically computed ones, and use the data stored under that context. Then, predict the overall ratings of the items. After that, the top N items from highest score are recommended to the user.

4. EXPERIMENTS

Two rounds of experiments are reported. The first round of experiments tests whether the prediction is better when context is considered compared to the prediction without considering context. And the second round of experiments tests whether the prediction is better when "borrowing" ratings from a similar context dimension, compared with "borrowing" ratings from a dissimilar dimension. Our experiments were implemented using MatLab 2007b. All the experiments were run based on a XP Professional based PC with Intel Pentium(R) 4, CPU 3.20GHz, and a 1 GB of RAM.

4.1 Dataset

A feedback and use ratings data for an example service domain (hotel stays) is collected from a well-known rankings site. The data is suitable since a large number of feedback rankings are available from a variety of sources, and our particular set had ratings of five individual criteria, one total rating and information about the context of use. The five criteria are from Value, Rooms, Location, Cleanliness and Service. All the individual criteria ratings and the total ratings ranged from 1 to 5, with 5 as the excellent.

Apart from criteria dimensions, there are also three context dimensions, DateofStay, VisitWasFor, and TraveledWith. Two values for VisitWasFor, leisure and business, while there are eight values for TraveledWith: "with spouse", "with friends", etc. Since the raw data is too sparse to compute, we aggregate ratings of all the hotels from the same brand with the same number of stars, using the assumed equality between the levels of service and the similarity of furnishing standards of such hotels. After aggregating, there are 225,991 records, with 8934 hotels and 178,863 users.

4.2 Contextual Structure

As mentioned in Section 3.4, when considering context, it is important to test which context dimension really affects rating estimations significantly. All the context dimensions in our dataset are categorical, so chi-square tests were applied for each context dimension and total ratings. After testing, on our data set, VisitWasFor and TraveledWith context dimensions showed the significant difference in estimating total ratings. Thus in the rest experiments, VisitWasFor and TraveledWith were considered as our context dimensions.

Then we used specialization taxonomy to build the structure of the two context dimensions. Figure 2 shows the hierarchical specialization structure of TraveledWith context. The eight circles with Bold and Italic characters are TraveledWith context values.

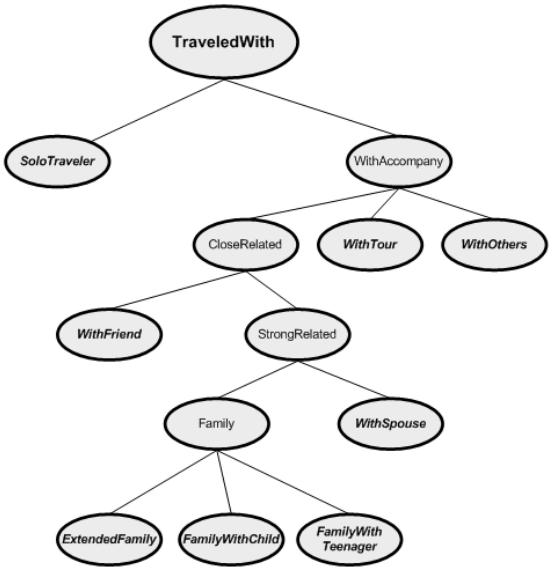

Figure 2. TraveledWith context structure

4.3 Experimental Design

The raw records were cleaned for further computation, as there were 151,964 users who only rated one hotel of the total 178,863 users. In the following experiments, users who had rated more than 3 hotels were considered. In order to get a less sparse user-hotel matrix, for the rest each record, which is defined by a user and a hotel, we calculated sum of the number of ratings for this user, and the number of ratings for this hotel. The sum was denoted as "survival score" of that record for the further selection. We experimented with different "survival score" thresholds to balance the number of remaining records whilst achieving a tolerable level of sparsity.

4.3.1 Effect of context consideration. In this first round of experiments we tested whether considering the context of use would produce better recommendation than recommendation without considering the context of use. In this first round we do experiments on two datasets. The first dataset is within one context entirely, and the other dataset spans different context. If the prediction result of the first dataset is better than the second dataset, then it

proves that context consideration will improve the prediction accuracy.

We use the context of VisitWasFor, with two main values "for Business" and "for Leisure". In our final dataset, the data under visiting for Business is far more than visiting for Leisure. To reduce the problems that can be caused by one context domination, we set two data sets with even number of records. 319 records with the highest survival scores are selected from context Business, stored in Dataset 1. Then for Dataset 2, all the 155 records from context Leisure are selected. And the other 164 records were selected from Dataset 1.

Then we divided our datasets into a training set and a test set. 90% of the data was used as training set, and 10% of the data was used as test set. The approach proposed in the paper was applied in the training set, and did the prediction for the users in the test set.

4.3.2 Effect of similar context prediction. In the second round of experiments we tested whether data from similar context can be "borrowed" to predict ratings from an "active" context with insufficient number of ratings, and more importantly, if these predictions are more accurate when compared to "borrowing" data from dissimilar context.

The dataset used in the second round of experiments demonstrated an expected issue where different context dimensions influence each other. In our example the VisitWasFor value of "for Business" only had records covering the TravelledWith segments of "Others" and "Solo", whilst the VisitWasFor value of Leisure had data under the TraveledWith segments of Others, Friend, Spouse, ExtendedFamily, and Solo. Therefore we used the data under "VisitWasFor" value of "Leisure" to conduct our further tests. In the dataset, the number of records under WithSpouse (856 records) is far more the number of other contexts. Then WithOthers (222 records) and WithFriend (147 records) follow. Solo (53 records) and ExtendedFamily (50 records) have a few records.

According to the features of our dataset and with limited number of data, in these experiments, we choose the data under ExtendedFamily as our test set. From the TraveledWith hierarchy tree above, we can see that ExtendedFamily is close to WithSpouse, but far from WithOthers, and Solo. For the purpose of testing if the distance between context concepts on the tree affect the prediction, WithFriend dataset was not considered because it is in the middle, and it may make the results obscure. Thus we choose WithSpouse as one of the training set, WithOthers and Solo as another training set. Then we compare the recommendation prediction by using different training sets to show which context predict more accurate.

Since the records in Spouse training set (856 records) are much more than those in Solo and Others training set (275 records), thus to keep the experiments reasonable, only 275 records should be left in Spouse training set. Under the principle, to keep the sparsity level of two training sets similar to each other, 275 records are left with a sparsity level 0.9606, while the sparsity level in Solo and Others training set is 0.9650. Sparsity level is defined as $1 - totalrecords / userNum * hotelNum$.

4.3.3 Evaluation metric. Mean Absolute Error (MAE) is used as the evaluation metric in this paper. MAE is a measure of the average absolute deviation between a predicted rating and the user's true rating.

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \qquad (9)$$

N is the number of pairs of real ratings and predictions $< p_i, q_i >$. The lower MAE, the more accurate predictions are [29].

4.4 Experiment Results

4.4.1 Impact of Context Dimensions. We ran our experiments on the proposed method in Section 3.4 using the setup detailed in Section 4.3.1, and predicted the score for each user on the hotel(s) in the test set. Then MAE was computed for both data sets. Figure 3 shows part of the total rating matrix. Rows are for hotels, while columns are for users.

	1	2	3	4	5	6	7
A	4	3	4		2		
B				3		3	
C	3		4				
D		2			2		
E							
F						4	
G							
H	3.75		3	4.5			5
I					4	4	4
J			3	4	4	3	

Figure 3. Part of total rating matrix under context Business

The experimental results are shown in Table 1. From the table, MAE within only one context dimension 0.6768, is significantly smaller than MAE across both context dimensions which is 0.8806. The result on Dataset 1 is a reasonable prediction within the constraints of the limited amount of data. In the future, we will experiment on a larger data set. The results prove our assumption that context affects users' ratings, and that the prediction within only one context dimension is better than the prediction across context dimensions. Therefore RS predictions which

consider context would be in the general case more accurate than predictions on the whole data set without considering the context information.

Table 1. MAE of both datasets

	DataSet 1	DataSet 2
No. of Records	319	319
MAE	0.6768	0.8806
Context Dimension	Business	Business & Leisure

4.4.2 Impact of similar context. We ran our experiments on the proposed method in Section 3.4, using the setup detailed in Section 4.3.2, and predicted the score for each user on the hotel(s) in the test set using different training sets. Then MAE was computed for these data sets.

Table 2. Experiment Results

Training Set	Records Num	Sparsity Level	MAE
Solo+Others	275	0.965	1.083
Spouse	275	0.9606	0.8543
Spouse	856	0.9769	0.3434

From the table above, we can see that when both the numbers of the two training sets and the sparsity level are similar, MAE of Spouse training set is smaller, with a value of 0.8543, which means Spouse training set provides a better prediction for the test set. From Figure 2, the contextual hierarchy tree, we can see that Spouse is closer to ExtendedFamily, than Solo and Others. The experiments support our assumption that the closer the two context dimensions are, the better prediction is. However, the values of MAE for both training sets are not impressive.

When we did the prediction on the original data set of Spouse, with 856 records, we obtained a very small MAE, 0.3434, which is much smaller than the prediction result under 275 records. The comparison indicates that the relatively high MAE of Spouse with 275 records is caused by the very limited number of records within each training set.

5. Conclusion & future work

We conclude that a service recommender system should consider a number of user-oriented criteria and multiple context dimensions, and that the field of multi-criteria decision analysis can offer useful ranking models and processes.

Our approach developed a method which covers both criteria and context dimensions. Euclidean Distance is used to compute the multiple criteria ratings of users on each item. For contextual information, reduction-based approach is introduced. However, for the sparsity problem caused by reduction-based approach, we use similar context to reduce its side effect. Concept specialization taxonomy is used to structure these context dimensions, and help us establish the distance and similarity between categorical context segments. Indeed, we need different similarity calculations for different types of context. For scale and ordinal types, we use the inverse of the normalized distance, so that the closer the values are, the higher the similarity measure and the more similar are the two contexts. For the categorical type, the specialization hierarchy tree will allow us to establish simple metrics such as the number of specialization links between the concepts, or the size of the concept difference (in description logic).

We have used experiments to establish that the proposal is sound, and that the recommendation using context segment information is better than recommendation which does not use context information. The second experiment validated the mechanism for tackling intra-context sparsity, by demonstrating that the "borrowing" of data from a very similar context produces better predictions than using the data from a remote context.

One of the weaknesses of the paper is that the similarity of categorical context is estimated using the simple measure of counting specialization edges, and is thus dependent on the degree of granularity of the concept taxonomy. However, we are planning to measure the similarity of two categorical context dimensions by using semantic reasoning in the future. Then the similarity will be used as weight for a better recommendation prediction. This area of multi-criteria and multidimensional recommender system research is to be developed further, and the way in which these models and processes are integrated within the recommender systems should be carefully considered within our further research plans.

6. References

[1] Burke, R., Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 2002. Volume 12(Number 4): p. 331-370.

[2] Schafer, J.B., J.A. Konstan, and J. Riedl. Recommender Systems in E-Commerce. in Proceedings of the 1st ACM conference on Electronic commerce 1999.

[3] Adomavicius, G. and A. Tuzhilin, Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE transactions on knowledge and data engineering, 2005. 17(6).

[4] Manouselis, N. and C. Costopoulou, Analysis and Classification of Multi-Criteria Recommender Systems World Wide Web, 2007. 10: p. 415-441.
[5] Adomavicius, G., et al., Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems (TOIS), 2005. 23 (1): p. 103 - 145
[6] Karta, K., An Investigation on Personalized Collaborative Filtering for Web Service Selection. 2005.
[7] Balabanović, M. and Y. Shoham, Fab: content-based, collaborative recommendation, in Communications of the ACM. 1997. p. 66 - 72
[8] Leimstoll, U. and H. Stormer. collaborative recommender systems for online shops. in Proceedings or the 13th Americas Conference on Information Systems. 2007.
[9] Huang, Z., W. Chung, and H. Chen, A Graph Model for E-Commerce Recommender Systems. Journal of The American Society for Information Science and Techonology, 2004. 55(3): p. 16.
[10] Anand, S.S. and B. Mobasher, Intelligent Techniques for Web Personalization in Intelligent Techniques for Web Personalization 2005, Springer Berlin / Heidelberg. p. 1-36.
[11] Resnick, P., et al. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. in Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work. 1994.
[12] Shardanand, U. and P. Maes. Social information filtering: algorithms for automating "word of mouth". in Proceedings of the SIGCHI conference on Human factors in computing systems. 1995.
[13] Breese, J.S., D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. in Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. 1998.
[14] Anand, S.S. and B. Mobasher, Contextual Recommendation in From Web to Social Web: Discovering and Deploying User and Content Profiles. 2007, Springer Berlin / Heidelberg. p. 142-160.
[15] Adomavicius, G. and A. Tuzhilin, Multidimensional Recommender Systems: A Data Warehousing Approach in Electronic Commerce. 2001, Springer Berlin / Heidelberg. p. 180-192.
[16] Nguyen, H. and P. Haddawy. DIVA: Applying Decision Theory to Collaborative Filtering. in Proceedings of the Conference on Artificial Intelligence for Electric Commerce. 1999.
[17] Lee, H.-H. and W.-G. Teng, Incorporating Multi-Criteria Ratings in Recommendation Systems, in IEEE International Conference on Information Reuse and Integration, 2007. 2007. p. 273-278.
[18] Adomavicius, G. and Y. Kwon, New Recommendation Techniques for Multicriteria Rating Systems, in IEEE Intelligent Systems. 2007. p. 48-55.
[19] Matsatsinis, N.F., K. Lakiotaki, and P. Delia. A system based on multiple criteria analysis for scientific paper recommendation. in Proceedings of the 11th Panhellenic Conference on Informatics. 2007.
[20] Lakiotaki, K., S. Tsafarakis, and N. Matsatsinis, UTA-Rec: A Recommender System based on Multiple Criteria Analysis, in The 2nd ACM conference of Recommender Systems. 2008. p. 219-225.
[21] Chen, A., Context-Aware Collaborative Filtering System: Predicting the User's Preference in the Ubiquitous Computing Environment, in Location- and Context-Awareness. 2005, Springer Berlin / Heidelberg. p. 244-253.
[22] Chapphannarungsri, K. and S. Maneeroj. Combining Multiple Criteria and Multidimension for Movie Recommender System. in Proceedings of the International Multiconference of Engineers and Computer Scientists. 2009.
[23] Palmisano, C., A. Tuzhilin, and M. Gorgoglione, Using Context to Improve Predictive Modeling of Customers in Personalization Applications. IEEE transactions on knowledge and data engineering, 2008. 20(11): p. 1535-1549.
[24] Schilit, B., N. Adams, and R. Want, Context-aware Computing Applications, in 1st International Workshop on Mobile Computing Systems and Applications. 1994. p. 85-90.
[25] Lieberman, H. and T. Selker, Out of Context: Computer Systems that Adapt to, and Learn from, Context. IBM Systems Journal, 2000. 39: p. 617-632.
[26] Dey, A.K., Understanding and Using Context Personal and Ubiquitous Computing, 2001. 5(1): p. 4-7.
[27] Lilien, G.L., P. Kotler, and S.K. Moonrthy, Marketing Models. 1992: Prentice Hall.
[28] Kwon, O. and M. Kim, MyMessage: Case-based Reasoning and Multicriteria Decision Making Techniques for Intelligent Context-aware Message Filtering. Expert Systems with Applications, 2004. 27: p. 467-480.
[29] Sarwar, B., et al. Item-based collaborative filtering recommendation algorithms. 2001. Proceedings of the 10th international conference on World Wide Web.