Semantic Content-based Recommendation of Software Services using Context

LIWEI LIU, University of Manchester
FREDDY LECUE, IBM Research
NIKOLAY MEHANDJIEV, University of Manchester

The current proliferation of software services means users should be supported when selecting one service out of the many which meet their needs. Recommender Systems provide such support for selecting products and conventional services, yet their direct application to software services is not straightforward, because of the current scarcity of available user feedback, and the need to fine-tune software services to the context of intended use. In this paper, we address these issues by proposing a semantic content-based recommendation approach which analyses the context of intended service use to provide effective recommendations in conditions of scarce user feedback. The paper ends with two experiments based on a realistic set of semantic services. The first experiment demonstrates how the proposed semantic content-based approach can produce effective recommendations using semantic reasoning over service specifications by comparing it with three other approaches. The second experiment demonstrates the effectiveness of the proposed context analysis mechanism by comparing the performance of both context-aware and plain versions of our semantic content-based approach, benchmarked against user-performed selection informed by context.

Categories and Subject Descriptors: **H.3.5 [Online Information Services]**: Web-based services; **H.3.3 [Information Storage and Retrieval]**: Information Search and Retrieval

General Terms: Performance, Experimentation
Additional Key Words and Phrases: Service Descriptions, Semantic Web Services, Recommender Systems, Context, Semantic Content-based Approach

ACM Reference Format:

Gang Zhou, Yafeng Wu, Ting Yan, Tian He, Chengdu Huang, John A. Stankovic, and Tarek F. Abdelzaher, 2010. A multi-frequency MAC specially designed for wireless sensor network applications. *ACM Trans. Embedd. Comput. Syst.* 9, 4, Article 39 (March 2010), 6 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Recommender Systems are seen as an effective solution to the problem of shoppers on e-Commerce sites such as Amazon.com and eBay having to choose one product from the multitude of those which satisfy their requirements [Schafer et al. 1999], [Schafer et al. 2001]. Traditionally, a recommender system would ask shoppers to rate the products they have bought or used. Based on these ratings, a recommender system would predict the rating a particular user would give to an item yet unknown to them. It thus recommends those items with the highest predicted ratings.

This paper is an extended version of a paper appearing in the 9th International Conference on Web Services [Liu et al. 2011]. It shares some similarity with another conference paper appearing in the 4th IEEE International Conference on Semantic Computing [Liu et al. 2010] as well.

Author's addresses: L. Liu and N. Mehandjiev, Centre of Service Research, the University of Manchester; F. Lecue, IBM Research, Smarter Cities Technology Centre, Dublin, Ireland.

DOI:http://dx.doi.org/10.1145/0000000.0000000

The growth of the different varieties of software services, such as web services [Papazoglou 2008], semantic services [McIlraith et al. 2001] and grid services [Goble and Roure 2002], means that software services are naturally seen as the next application area for recommender systems. The concept of a software service focuses on its "on-demand" nature lead by user needs: "services are configured to meet a specific set of requirements at a point in time, executed and discarded" [Bennett et al. 2000]. In this paper, we use web services and semantic services as well-known instantiations of this concept.

The increased needs for standardization of distributed computation, and the "democratization" of the Internet of Services, allow a multitude of users with varied skills and abilities to assemble mashups and other service-related applications to support both business processes and leisure activities [Liu et al. 2010]. The multiplicity of available services increases the needs for supporting service consumers and service integrators in the selection of services which are the ones best matching their needs. Apart from the desired service functionality, a powerful determinant of the appropriateness of a service is the similarity between the context for which it has been developed and the context within which it is intended to be used, since the consideration of context can render the whole process more user-centered. For example a music download service developed for use in the United Kingdom would employ a number of assumptions about currency, copyright regulations and applicability of VAT to any sales. These assumptions will not hold in other countries, and using this service elsewhere may be even illegal, despite the apparent match at the level of functional specifications of the service. The full definition of context is discussed in Section 3.2.

Existing recommender systems need to be adjusted for service selection, because of the differences in *characteristics* and *marketplace maturity* between software services and products. The *characteristics* which differentiate the selection of software services compared to the selection of products are rooted in the greater involvement of the service consumer in the delivery of a service [Sampson and Froehle 2006]. Compared with product usage, service episodes are richer and more context sensitive [Sreenath and Singh 2004], which puts additional concern on contextual information of services. In terms of *marketplace maturity*, for example, software service marketplaces and search portals such as ProgrammableWeb[1], XMethods[2], WebServiceList[3] and Seekda[4] are currently immature and do not have the wealth of user feedback, reviews and rankings which characterize their mature counterparts focused on products (i.e. PriceRunner[5], Ciao[6]) or conventional services (i.e. TripAdvisor[7]). This lack of user feedback is also known as the "cold-start [Adomavicius and Tuzhilin 2005]. To address this problem we propose a content-based recommendation technique using semantic similarity measures, which we have extended to take into consideration the context information associated with service use.

In this paper, we focus on the semantic content-based approach as more appropriate to the current immature state of the service marketplaces, and show how

[1] http://www.programmableweb.com/
[2] http://www.xmethods.com
[3] http://www.webservicelist.com/
[4] http://webservices.seekda.com/
[5] http://www.pricerunner.co.uk/
[6] http://www.ciao.co.uk/
[7] http://www.tripadvisor.co.uk/

this can be integrated with context-aware recommendation which makes the recommendation more aligned with the intended context of use. Our semantic content-based approach, the first contribution in this paper, matches services through five different aspects, input, output, precondition, effect and functional category. It filters services with different functionalities, and also can distinguish between services with similar functional properties but different service categories. Our context-aware recommendation, which is the second contribution of this paper, calculates the similarity of context segments, so that it can select services which are suitable for the target context of use, as identified through either users' purchase history or service providers' specifications. Both contributions are based on a semantic similarity measure, which provides a good basis for their precision and integration as we demonstrate with our experiments.

The rest of the paper is structured as follows: we start by reviewing existing work in the area of recommender systems for software services. Then in Section 3 we introduce the necessary background and also specify the concepts of semantic web service and context as they are used in the paper. In Section 4, we provide an overview of our approach, describe how the technique for semantic similarity measure underpins content-based recommendation and context analysis, and also provide the details about combining them together in one approach. In Section 5, we test our approach using a realistic data set of semantic services, comparing its recommendations against alternative approaches to semantic reasoning on web services, before concluding the paper in Section 6.

2. STATE OF ART

2.1 Recommending Services

As mentioned in the Introduction, recommender systems offer items which a user can find of interest, based on their previous history or purchases or rankings. The recommendation approaches are usually classified into three categories: collaborative filtering (CF), content-based and hybrid approaches. CF approaches recommend items to a user based on the history of users identified as similar to him or her. Content-based approaches recommend items similar to those the user liked in the past, based on the item's specifications or descriptions [Adomavicius et al. 2005], [Adomavicius and Tuzhilin 2005].

The work on recommending software services has so far focused on directly applying classical recommendation approaches to software service recommendations. An example of this is applying CF approach. [Manikrao and Prabhakar 2005] design a framework that finds services through measuring the semantic distance between the conceptual annotations associated with service attributes, and then it asks the user to give a rating about the software services they have executed and calculate services similarity by subtracting the average ratings of services. A CF approach is used to predict the ratings of services that the user potentially selects. Zheng et al. [2009] produce a CF approach for recommending web services under the assumption of the existence of users' rating feedback on different QoS attributes of services. Their approach combines both user-based and item-based similarity to overcome the lack of either users or web services to compute the similarity. However, the CF approach suffers from "cold-start" problem, where, at the beginning, not much user feedback is available, and the monitored QoS values have limited ability to present the real users' feedback on those attributes.

Besides using CF approaches, content-based approaches for recommending software services also exist. For example, Blake and Nowlan [2007] introduce a recommender system to suggest web services for users' daily routines. The system creates an approach that combine the nature of message naming and standard string manipulation approaches. At the same time, the strings in user profiles are examined to find how similar they are to the names of inputs and outputs of web services. Recommendation is given based on the similarity. They actually employ an enhanced syntactical method, employing distances such as Levenshtein Distance [Navarro 2001] and Letter Paring to compare syntactic contents. Alternatively, Bouquet et al [2005] use ontological distance between concepts i.e., the length of the minimal path between the concept nodes corresponding in the ontology. Kaufer and Klusch [2007] implement a service matchmaker WSMO-MX, which retrieves and ranks services based on logic-based and text similarity filters for a given query. The rankings are computed by aggregated valuations based on ontology-based type matching, logical constraint matching, relation name matching and syntactic similarity measurement [Kaufer and Klusch 2007].

The research area of semantic computing can bring software services to their full potential [Terziyan and Kononenko 2003], such as Description Logics (DLs) reasoning [Cohen et al. 1992; Horrocks 1998; Küsters 2001] and semantic similarity measures [Cordì et al. 2005]. Some work has been done in incorporating semantic reasoning with recommender systems. Klusch and Kapahnke [2010] implement a semantic service matchmaker called iSeM. It provides (a) approximated logical signature matching based on non-monotonic concept abduction and information-theoretic similarity, (b) stateless strict logical specification, (c) SVM (support vector machine)-based semantic relevance learning with the full functional service profile matching and approximated logical signature matching. The preliminary results indicate that adaptive hybrid combination performs the best among them [Klusch and Kapahnke 2010].

2.2 Using Context

The approaches covered so far fail to consider the contextual information. The importance of contextual information in recommendation is rooted in the potential improvement in recommendation by considering a number of contextual factors, including the state of user, physical environment. The majority of research in context-aware systems for web service discovery and recommendation has focused on mobile phone infrastructure, e.g., [Pashtan et al. 2003], [Debaty et al. 2005] *etc*. Their concerns with physical context environment and pervasive sensors remain outside the focus of this paper where we consider a more abstract view of context and presume conventional IT infrastructure.

The recommendation is demonstrated to become more accurate when contextual information is considered [Adomavicius et al. 2005]. Introducing context changes recommendation problem from the two-dimensional, where the recommendation is based on the dimensions of user and item's content (or description), represented as $User \times Content$ to the three-dimensional $User \times Content \times Context$. Indeed, a significant volume of work has been focusing on *extended recommender systems* which go beyond the traditional two dimensions, including those covering context dependent recommendations. One of the approaches to dealing with multiple contextual dimensions in recommender system is called the *reduction-based approach* [Adomavicius et al. 2005]. It reduces a multidimensional recommendations problem to a two-dimensional problem. For example, the prediction in a three

dimensional space $S = User \times Content \times Context$, can be expressed as the prediction in a two dimensional space $S' = User \times Content$ where $Context = c'$ [Adomavicius et al. 2005]. Thus the three dimensions are converted into traditional two dimensions within a particular context. The simplicity of this approach comes at a price, which in this case is the problem of insufficient data. Indeed once we limit the applicable set of ratings to only those provided within a certain set of context values, their number will in many cases be too small to underpin reliable recommendation. We address this problem through our semantic context similarity mechanism.

2.3 Overlaps with Service Discovery Work

Often discovery and recommendation are not clearly distinguished. Discovery takes as input a query that describes the current user interests, while recommendation research typically does not use an explicit query but rather analyzes the available user profile and context [Garcia-Molina et al. 2011]. In terms of functionality, discovery is mainly about finding a list of web services that meet the specified functional service description, while recommendation, which also includes selection, deals with choosing the most suitable web services from the list. In other words, discovery is a prerequisite for selection [Sreenath and Singh 2004]. However, they do share some similarity with the aim of "matching a context (which may or may not include an explicit query) to a collection of information objects", and their distinction has been blurred due to their extension towards personalization [Garcia-Molina et al. 2011]. What is more, a big amount of work that covers discovery also includes the sorting of the discovery results. Generally, services tend to be recommended following being discovered. Referring to the difference between service selection and recommendation, the selection process includes discovery and recommendation in [Manikrao and Prabhakar 2005]. In the other words, recommendation is part of part of selection framework. However, in [Sreenath and Singh 2004], selection is defined as involving "mapping a set of services to a service", and in a general form, it is mapping "a set of services to a ranking of the services in that set" [Sreenath and Singh 2004]. We tend to agree with the later, and consider the papers on discovery and recommendation to have many similarities focused on their shared concern with ranking items. This overlap allows us to consider work on context from within the service discovery area, and indeed, there are recent efforts in this direction. As an example, Dietze et al [2010] present a framework where discovery is performed according to a given situational context. The situation represents a specific state of the world and context is considered to provide more appropriate services. Our work extends the latter by integrating context for service recommendation following finer grained semantic techniques.

2.4 Content and Context-based Approaches

There is some work done in combining content-based approach with context analysis. For instance, Segev and Toch [2009] propose a context-based semantic approach to match and rank web services. In the paper, the context is used to describe the related set of linguistic terms of a given text. They extract tokens which are the textual terms by parsing the underlying documentation of services, and use string-matching function to match ontologies of those tokens. Broens et al [2004] present a service discovery approach by matching user's query and service description, and associated contextual information. In their approach, contextual information which is provided by context providers, service descriptions which are provided by service providers and service request which is provided by users, all these three will be modeled by

ontology. Then the three pieces of information are matched one after another [Broens et al. 2004]. Medjahed and Atif [2007] propose a web service contexts categorization and then use ontology to define this categorization. The contexts are modeled by a two-level mechanism which covers both context specifications and service policies. A peer-to-peer architecture is presented to fully match web services context policies. Each context of a source service is matched by a policy of the candidate service. And rule-based techniques are used for comparing context policies. In our paper, we use a different notion of context from the papers we mentioned above as discussed in Section 3.2. In addition, instead of doing semantic reasoning only on the content descriptions of services, we also apply reasoning to their contextual information in order to gain a more accurate result.

3. CRITERIA FOR SOFTWARE SERVICE RECOMMENDATION

In this section we first provide some background regarding semantically annotated software services and contextual information, and then we focus on the specific features relevant to this paper.

3.1 Software Services

In our approach software services are described using semantic descriptions, so in this section we focus on semantic specifications of software services which can be used in a content-based recommender system.

Here we review 1) semantic service descriptions, and 2) a non standard DL reasoning techniques used by us to infer the commonality and differences in service descriptions.

3.1.1 Semantic Descriptions of Software Services. The formal model required to represent semantics of a software 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 $\mathcal{E}(s)$;

All are provided by a domain ontology $\mathcal{T}$ through semantic annotations. The particular ontology $\mathcal{T}$ is based on the DL $\mathcal{ALE}$ [Baader and Nutt 2003], mainly defined by its Terminological Box (or TBox i.e., intentional knowledge) in DL In the following, the TBox is used to annotate service descriptions, and it also supports inference on these descriptions by means of DL reasoning. For the sake of clarity, we assume services without open preconditions. Figure 1 shows a fragment of an example TBox $\mathcal{T}$.

According to this model, semantic software 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 OWL-S profile [Ankolenkar et al. 2004], WSMO capability [Fensel et al. 2005] or SA-WSDL [Sivashanmugam et al. 2003] (formerly WSDL-S) can be used to describe these parameters including the functional category.

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

Fig. 1. Part of an $\mathcal{ALE}$ (a tractable family of DLs) TBox

Example 1: (A Semantic software service). Suppose a software service s_1 with its semantic description in the TBox $\mathcal{T}$. s_1 is defined with $FootwearProductDescription$ as functional category, which returns a list of information $ProductData$ (e.g., Name, Origin) of a product identity $ProductID$ in a given $Format$. In addition the service required, as a precondition, the input parameter $ProductID$ to be valid. Such a service does not have any effects.

3.1.2 Common and Missing Description. Given the definition of semantic software service, recommender systems 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 and sd_j (referring to any attribute of service descriptions), encoded using the same TBox $\mathcal{T}$, can be judged using a matchmaking function. This function enables finding some (basic) levels of semantic compatibilities [Li and Horrocks 2003; Noia et al. 2003; Paolucci et al. 2002] (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 a more detailed information i.e., the DL concept descriptions: *Missing* and *Common Descriptions* (first defined as the *Extra* and *Common Descriptions* in [Lécué and Delteil 2007].

On the one hand the computation of *Missing Descriptions* is done by exploiting a non-standard DL reasoning: the *difference* or subtraction operation [Brandt et al. 2002] for comparing $\mathcal{ALE}$ DL-based descriptions, thus obtaining a compact representation of the metric:
(i) the Missing Description $sd_j \backslash sd_i$

$$sd_j \backslash sd_i \doteq \min_{\preccurlyeq d}\{E | E \sqcap sd_i \equiv sd_j \sqcap sd_i\} \tag{1}$$

which refers, with respect to the subdescription ordering $\preccurlyeq d$ [Küsters 2001], 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 $T \vDash sd_i \sqsupseteq sd_j$, Equation (1) refers to information which is required by sd_i to be

similar sd_j. The *Missing Description* 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 [Cohen et al. 1992] lcs as a DL concept description), i.e.,

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

which refers to information shared by sd_i and sd_j.

Example 2: (Common and Missing Description). The description missing in $Product$ to be similar to $ProductData$ is referred by $ProductData \backslash Product$, due to subsumption, it can be written as $\forall hasPrice.ProductPrice \sqcap \exists hasPrice.ProductPrice$. The common Description of the latter descriptions is defined by their Least Common Subsumer, which refers to the information shared by $ProductData$ and the description $Product$ i.e., $lcs(Product, ProductData)$ i.e. $Product$. In other words, both descriptions are $Product$.

The DL intersection between the description Product and the Missing Description $ProductData \backslash Product$ i.e., $\forall hasPrice.ProductPrice \sqcap \exists hasPrice.ProductPrice$ is of Exact matching type with $ProductData$ i.e., perfect semantic similarity.

3.2 Context

As we have described, context plays a more and more important role in recommendation. However, there are no commonly accepted standard representations, models or ontology for contextual information till now. One part of the reasons is that context may include almost everything, and there is no clear definition and notion for context [Kocaballi and Kocyigit 2007]. From the literature, different researchers define context in different ways, and the usage of context differ from application to application as well. Besides the general definitions of context, researchers specify context variously in their applications for web services as well, such as in [Maamar et al. 2005], [Medjahed and Atif 2007] *etc.*

In our application, we present the context as the set of external conditions and circumstances for which or where the user intends to use a service. It is independent of the content of services (service description). It is more about the description of the potential interactions, as what Brezillon [2003] has defined, context is "the information that characterizes the interactions between humans, applications, and the environment" [Brezillon 2003]. In general, there are implicit relationships among user, service and context, since one of the basic assumptions in context-aware recommendation is that the user's selection on services changes under different context. There are exceptions where a service can be used in all kinds of context, and then this service is viewed as a *context-free* service.

If we present the information relevant to service recommendation along different dimensions, user's profile will be covered under the user dimension, whilst service description is included under service dimension which is used for matching services in our approach. Service's functional category, input, output, precondition and effect all of these information is included in service description, as expressed in the Section 3.1. The contextual information belongs to context dimension.

Examples of relevant context parameters in this paper are the *time of use* (in the morning or afternoon, weekday or weekend), *physical locations* (meeting room, cafeteria) [Maamar et al. 2006], *execution platform* (handheld device e.g. Mobile, fixed device e.g. computer), *intended use* (*directly use* or *use for service compositions*), *usage frequency* (daily use/High, Medium, one-off/Low), *type of use* (use for leisure or for work or other situation), and a manner of *service delivery* (computation results only or need for delivering the service code) and so on.

The mechanism for considering context proposed as one of the main contributions in this paper is not dependent on the number of context types chosen. To simplify the description without loss of generality, here we consider four pieces of contextual information which can generalize normal situation: *execution platform, intended use*, *usage frequency* and *type of use*. These four represent the wider set of parameters which can affect the selection of software services for a particular user. For example, for *usage frequency* type of context, if a service is to be used by a company, the expectation of having to process large volumes of data is potentially much stronger than if it is used by an individual. Thus, the requirements on execution time and capacity are higher. *Intended use* is another context dimension. *Direct use* and *used for composition* may have different levels of requirements on the availability of software services. And the values within each of these context dimensions, such as *directly use* within *intended use* context, are called *context values*. A context segment is a combination of values of attributes of one or more contextual dimensions, and each segment defines a subset of ratings related to it, following a segment definition from Adomavicius et al [2005]. For instance, $< leisure, handheld\ device >$ context segment contains all the software services under the *use for leisure* context value and *handheld* platform context value, and these two context values are from two context dimensions, *type of use* (*T* for short) and *execution platform* (*P* for short).

4. PROPOSED APPROACH

In this section, we start by explaining how our approach makes use of semantic similarity and context based information, and we illustrate how it works through a running example. Finally, we explain how to combine service-based and context-based recommendations.

4.1 Overview of the Approach

There are two main types of input data for our approach: semantic-based functional description of software services and semantic-based description of contextual information. Both types of data can be inferred from past interactions, or explicitly specified by the user when they search for a service. Service providers provide their services with service description, and they can also explicitly tag a service with context information, when a service has been developed for some specific context, since we believe that the service provider knows well how their services should be used and under which circumstance the services are suitable to use. The services which are not attached to any context segment, are defined as context-free services.

In general, our approach returns a list of services that are the closest match to both types of data, semantic description of services and contextual information. When a user is searching some specific functioned service, a list of services matching the functionality is returned through our semantic content-based approach, with the most frequent referred context information from those services. At the same time, this user is asked to specify the relevant context values from the returned context information. When the user is not selecting the context information, the context

setting then is parsed from his/her profile, which was provided by the user initially. Afterwards, there is a matching process between the user's context with the returned services attached context. The services from the same context or close context are taken as the final recommendation. As shown in [Liu et al. 2011] [Liu et al. 2010], services from similar context segments can be considered a good match when there is insufficient number of similar services from the same context segment.

For context-free services, only semantic descriptions of services are involved in the recommendation process. The matching process is made by semantic reasoning which estimates content similarity and separately, context similarity, as detailed in Section 4.2. The two are then combined using weighting ratio which can be set by the user. As common in recommendations when implicitly collecting users' feedback, we assume that a user likes the service s/he has selected or bought [Anand and Mobasher, 2005], and the default rating for that service is defined as 1 here. No further detailed ratings which present user's preferences after using the service are collected. An example of service descriptions of toys is given below to explain how the approach works.

Example 3: (Toy Description Services). A user searches for a software service which has the function of describing toy products. In the returned list of services within this functional category, there are $ProductDescription$ service (context-free), $FootwearProductDescription$ service (associated with the $< CompositionUse, HighFrequency, ForWork >$), $DollsToyDescription$ service (associated with $< DirectlyUse, LowFrequency, ForLeisure >$), $IntelligentRobotsDescription$ service (associated with $< DirectlyUse, HighFrequency, ForWork >$), $RobotsToyDescription$ service (associated with $< DirectlyUse, LowFrequency, ForLeisure >$), $Electronic-RideOnToyDescription$ service (associated with $< DirectlyUse, LowFrequency, For-Leisure >$) and so on. The context segment specified by the user is $< DirectlyUse, LowFrequency, ForLeisure >$. If the user is clicking a service called $RobotsToyDescription$ (or has used it in the past visits to our service catalogue), we know this user is looking for a service under the toy category. Then we match this service with the rest of the services in the list from 5 aspects, functional category, input, output, precondition and effect as described before. In the meantime, the context associated with the user (by virtue of previous interactions or explicit specification) will be matched with the other contextual information attached to the services through semantic reasoning as well. In this example, $ElectronicRide-OnToyDescription$ and $DollsToyDescription$ services will be returned with high scores, since this service is from the toy category, and with a perfect match on the context cells as well. The process is as shown in Figure 2.

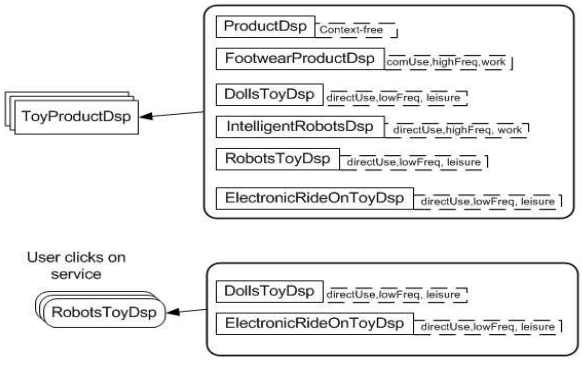

Fig. 2. Toy Description Services

4.2 Semantic Similarity

We introduce semantic similarity of software services for both extending traditional content-based approach with semantic measure of content similarity and for establishing similarity between segments with which a service is tagged. One generic measure is considered for evaluating semantic similarity between services descriptions: their Common Description rate.

Definition 4.1 (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:

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

This rate estimates the proportions of descriptions in sd_i and sd_j which are in common (the higher the proportion, the better the similarity). The expressions in between $||$ refer to the size of $\mathcal{ALE}$ concept descriptions ([Küsters 2001] p.17) i.e., $|T|$,$|\bot|$, $|A|$, $|\neg A|$and $|\exists r|$is 1; $|C \sqcap D| \doteq |C| + |D|$; $|\forall r.C|$ and $|\exists r.C|$ is $1 + |C|$. For instance $|ProductData|$ is 22 with respect to Figure 1.

Example 4: (Common Description rate). Suppose there are two services s_1 and s_2, as shown in Table 1. According to Equation (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., $q_{cd}(Product, ProductData)$. Such a rate informs about the similarity between two descriptions i.e., the higher this rate (i.e., the closer the descriptions in an ontology) the more similar are the descriptions.

$$\frac{|lcs(Product, ProductData)|}{|ProductData \backslash Product| + |lcs(Product, ProductData)|}$$

$$= \frac{|Product|}{|\forall hasPrice.ProductPrice \sqcap \exists hasPrice.ProductPrice| + |Product|} \tag{4}$$

$$= \frac{4}{5}$$

The common description rate is pre-computed through calculating the size of $\mathcal{ALE}$ concept descriptions, as described above. And it is provided through DL reasoning, the detail can be found in [Lécué and Delteil 2007].

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 $Out(s_j)$; $\mathcal{E}(s_i)$ and $\mathcal{E}(s_j)$; $P(s_i)$ and $P(s_j)$; $F(s_i)$ and $F(s_j)$ 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., $In(s_i)$ and $In(s_j)$ in Table I, the value of each quality criterion is retrieved by computing their average.

Table I. Semantic service description

Semantic Web Service s_i		s_1	s_2
Functional Category		Footwear ProductDescription	ProductDescription
Functional Parameters $F(s_i)$	$In(s_i)$	ProductID	ProductID
		Format	Format
	$Out(s_i)$	ProductData	Product
Requirements	$P(s_i)$	Valid	Valid
	$\mathcal{E}(s_i)$	None	None

In more complex cases, where the number of semantic descriptions are different between attributes of services, only comparable (in term of subsumption) pairs of descriptions are considered.

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

$$q(s_i, s_j) \doteq \sum_{l \in \{F, In, Out, P, \mathcal{E}\}} w_l \times q(l(s_i), l(s_j)) \tag{5}$$

Where $w_l \in (0,1]$ is the weight assigned to the lth service description attribute and $\sum_{l \in \{F,In,Out,P,\mathcal{E}\}} w_l = 1$. In this way a desired service attribute can be given a higher importance by simply adjusting w_l e.g., the functional category of a service could be weighted higher. Finally, the results returned by Equation (5) is a value in (0,1] referring to the common description rate between service s_i and service s_j.

Example 5: (Semantic Similarity of Services). Suppose two services s_i and s_j (see Table I). According to the previous example, we obtained $q_{cd}(Out(s_i), Out(s_j))$. The other quality of services descriptions attributes are computed using the same process along In, F, P and $\mathcal{E}$ (see Table II). Finally by means of Equation (5), we obtain:

$$q(s_1, s_2) \doteq \frac{43}{50} \tag{6}$$

Where the weight w_l is $1/5$ for each attribute.

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 $q(s_i, s_k)$.

Table II. Quality along different attributes

$q(l(s_i), l(s_j))$	$q_m(l(s_i), l(s_j))$
$q(F(s_1), F(s_2))$	1/2
$q(In(s_1), In(s_2))$	1
$q(Out(s_1), Out(s_2))$	4/5
$q(P(s_1), P(s_2))$	1
$q(\mathcal{E}(s_1), \mathcal{E}(s_2))$	1

4.3 Coupling Semantic-Content and Context Analysis for Software Service recommender system

In this section, we are presenting the detail of how the semantic-content approach and context analysis work. Suppose service s_1 is the service for which we are going to generate recommendations for.

4.3.1 Semantic content-based approach. Starting from the cold-start problem and taking the advantage of software services characteristics, our content-based approach is based on the semantic representation of software services. All the services are described through their semantic attributes of input, output, precondition, effect and functional category (see Section 3.1) as suggested by existing semantic annotation standards [Ankolenkar et al. 2004], [Fensel et al. 2005], [Sivashanmugam et al. 2003]. Based on their annotation, the similarity of any two semantically described web services can be computed through Equations (3) and (5), the detailed process is described in Section 4.2. Some existing approaches [Kaufer and Klusch 2007] [Klusch and Kapahnke 2010] do not reflect how descriptions could be dissimilar, whilst the Equation (3) captures this aspect by computing missing descriptions when comparing services. One of the advantages of this approach is that the semantic similarity between one software service and other software services can be precomputed offline. The online component is responsible for looking up similar services for the user's queries (implicit or explicit) using the precomputed values. Our approach will therefore scale up independently of the size of services on the website or the total number of users. It is only dependent on the number of the user queries. As a result, it can be fast on a very large set of services. Another benefit of our semantic reasoning approach is the inclusion of all five standard semantic aspects of services. Matching input and output of one service to the input and output of other services can filter out services with different processing while the matching between functional categories can separate the services even when their processing is similar. The top N of the most similar services can then be listed.

4.3.2 Context analysis. Each service is annotated with context values for which it is suitable. And these context values can be assigned by service providers, or inferred by successful uses of the service.

Example 6: (Context-Annotated Services). Suppose the $FootwearProductDescription$ service (associated with the $< CompositionUse, HighFrequency, ForWork >$), in Example 3. This context information is captured using a RDF representation in Figure 3. The context extension of the service, also in RDF, is captured by Figure 4. Note that an extra OWL description, defined in ontology "contextOnto" (in Figure 3) is used for capturing semantic representation of each piece of the contextual information – see context taxonomy in Figure 5).

```
@prefix contextOnto: <http://coconut.tie.nl:8080/.../context/ontology#> .

: IntendedUseContext    a contextOnto:Context;
      kpionto:hasContextType contextOnto:CompositionUse .
: UsageFrequenceContext     a contextOnto:Context;
      kpionto:hasContextType contextOnto:HighFrequencey .
: TypeOfUseContext          a contextOnto:Context;
      kpionto:hasContextType contextOnto:ForWork .
```

Fig. 3 Part of a Context Domain Instance

```
@prefix contextDomainInstance: <http://130.88.6.246/servicedesign/.../domain#> .
@prefix contextOnto: <http://130.88.6.246/servicedesign/.../context/ontology#> .
@prefix rdfs: http://www.w3.org/2000/01/rdf-schema# .
@prefix  sawsdl:  <http://www.w3.org/ns/sawsdl#> .
@prefix  wsl:  <http://www.wsmo.org/ns/wsmo-lite#> .

FootwearProductDescriptionService      a wsl:Service;
        rdfs:isDefinedBy<http:// 130.88.6.246/.../Services/FootwearProductDescription?wsdl>;
        sawsdl:modelReference domain:ProductDescriptionService,
        wsl:hasOperation :GetFootwearProductDescriptionOP;
        contextOnto:hasServiceContext
                contextDomainInstance:IntendedUseContext,
                contextDomainInstance:UsageFrequenceContext,
                contextDomainInstance: TypeOfUseContext .
```

Fig. 4 RDF Representation of a Context Extension for Semantic Web Service

When a user is searching for service s_1, the context segment in which s/he intends to use the service is elicited by either observing the past user selection, or explicitly asking him/her to specify the relevant values. When the user downloads a particular chosen service, this context segment will be stored in this user's profile and used to indicate this user's future context settings. When there are multiple context dimensions available from this user's profile, these context dimensions are matched against the ones attached to the returned services from searching process before asking the user to confirm his/her context setting.

At the same time, similar services can be attached to multiple context segments since the services and their attached context information are provided by different service providers or enclosed as suitable for their context by different users. Among these multiple context segments, the relevant context segment for each pair service-user is identified by applying matching functions [Li and Horrocks 2003; Noia et al. 2003; Paolucci et al. 2002] (Section 3.1.2) on context of usage (i.e., from the service perspective) and expected use (from the user perspective).

If services with the same functionality are associated with all context segments, this type of services can be seen as context-free. Context-free services are important when the attached context of candidate services for service s_1 is very different from user's context needs. In this paper, context information is modeled as a hierarchy tree here for clarity. The hierarchy tree of some of our illustrated context types is structured as shown in Figure 5.

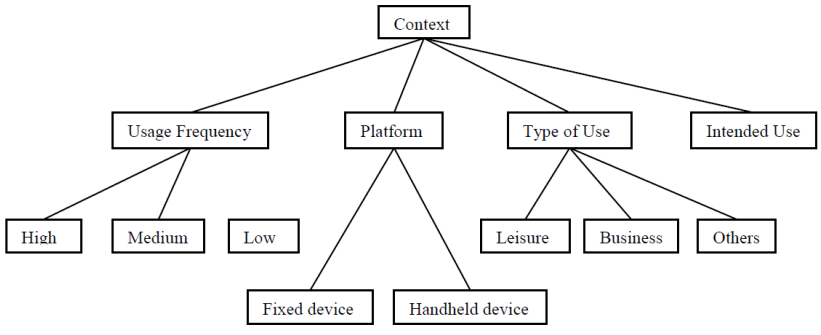

Fig. 5. Context Taxonomy

All the context dimensions are described by their domain ontologies through semantic annotations. Figure 6 is a sample of an $\mathcal{ALE}$ DL (E-Platform) terminology $\mathcal{O}$ for *Execute Platform* context dimension.

$$ExecutePlatform \equiv \forall IsPlatform.Platform \sqcap \exists IsPlatform.Platform$$
$$\equiv \forall IsFixed.ExecutePlatform \sqcap \exists IsFixed.ExecutePlatform$$
$$\sqcap \forall IsMoving.ExecutePlatform \sqcap \exists IsMoving.ExecutePlatform$$
$$FixedDevice \equiv \forall IsFixed.ExecutePlatform \sqcap \exists IsFixed.ExecutePlatform$$
$$HandheldDevice \equiv \forall IsMoving.ExecutePlatform \sqcap \exists IsMoving.ExecutePlatform$$
$$Computer \equiv \forall IsPlatform.FixedDevice \sqcap \exists IsPlatform.FixedDevice$$
$$Phone \equiv \forall IsPlatform.HandheldDevice \sqcap \exists IsPlatform.HandheldDevice$$
$$Platform \sqsubset \top$$

Fig. 6. Sample of an $\mathcal{ALE}$ Terminology T for Platform context

In other words, we model the context types in a DL ontology. The similarity will be computed by measuring the semantic similarity from an active context to a substitute context using their Common Description rate in Equation (3) as mentioned Section 4.2. In the formula, service representations sd_i and sd_j are replaced by context values within context segments. For instance, in calculating the similarity between context segment $< ForLeisure, HandHeldDevice >$ and context segment $< ForBusiness, FixedDevice >$, $ForLeisure$ and $HandheldDevice$ can be seen as two concepts within a service representation (e.g., input) of service s_1, while $ForBusiness$ and $FixedDevice$ can be seen as two concepts within the service representation (e.g., input) of another service, say s_2. In this case, the calculation of the similarity between two context segments becomes the calculation of the similarity between two service representations (e.g., inputs) of two services s_1 and s_2.

4.3.3 Recommendation Generation. After the above two steps, we can get all the similarities between given service s_1 and all the other services on the alternative list, and also the similarities of the "context of use" segments. The final prediction score is based on both content similarity and context similarity. Users are able to decide the weights of the two dimensions. To recommend for service s_i, its similarity to another service s_j can be calculated through Equation (7).

$$SimTotal(s_i, s_j) = w_s \cdot SimService(s_i, s_j) + w_c \cdot SimContext(s_i, s_j) \quad (7)$$

$SimTotal(s_i, s_j)$ is the final similarity we obtained between service $s_i$and s_j, which ranges in [0,1]. w_s is the weight that the user gives to the content dimension, and w_c is the weight given to context dimension, $w_s + w_c = 1$. For context-free services, both $SimContext_{ij}$ and w_c are 0.

The similarity between services content is taken as a priority. Only the services matching the user's queries are taken into consideration for the step of context matching. In general, the recommendations are generated on the principle that the higher the overall similarity with given service based both on the semantic content information and on the intended use of contextual information, the higher the chance for this service to be needed. We can either predict the overall similarity for a service to a user by selecting the highest similarity from all the estimated similarities. Alternatively, we can recommend the N best software services to a user or a set of users to a service.

5. IMPLEMENTATION DETAILS AND EXPERIMENTATION

We have implemented our semantic reasoning content-based approach in Java within Eclipse IDE. The experiments have been conducted on Intel(R) Core(TM) 2 CPU, 2.4GHz and 2GB RAM. In this paper, we evaluate the performances of our semantic content-based approach by comparing it with a hybrid semantic matchmaker, called iSeM [Klusch and Kapahnke 2010] in terms of precision, recall and F1 [Klusch et al. 2010]. To evaluate the performances of its effectiveness of processing contextual information, we compare the ranking provided by our approach with the rankings from users who are with knowledge of web services in the condition of considering services attached contextual information in terms of precision, recall and F1.

5.1 Implementation Details

We are using test collection SAWSDL-TC1[8]. It is semi-automatically derived from OWLS-TC 2.2 using the OWLS2WSDL[9] tool, which transforms OWL-S service descriptions (and concept definitions relevant for parameter description) to WSDL through syntactic transformation [Klusch et al. 2010]. It provides around 900 semantic web service descriptions written in SAWSDL (for WSDL 1.1) from 7 domains (education, economy *etc*). This test collection also comprises a set of queries and binary relevance sets specified by domain experts. Each service contains only a single interface with one operation. The inputs and outputs of each service are annotated using concepts described in OWL-DL exclusively. The concepts used in the service input and output refer to ontologies which are described in OWL [Klusch et 2010].

Our reasoning on the semantic similarity of two semantic web services, and their semantically described context segments is calculated by the common description rate. This common description rate is obtained by computing the Missing Description (through an adaptation of Fact++ [Horrocks 1998] with DL difference), the Common Description and the size [Küsters 2001] of DL concepts. The reasoning process was built on the WSMO [Bruijn et al. 2005] or WSML described files.

There are limited datasets with a relatively big number of semantic software services available, especially the ones that are described by WSMO. The software services within the test collection SAWSDL-TC1 used in this work are described by WSDL. In order to reason over these semantic software services within our reasoner, we manually converted 57 services from three relevance sets from SAWSDL-TC1. From the query set of the test collection, four queries to service are selected as user inputs to recommendation systems. A transformed service with its intended use contextual information example described with WSML is shown as Figure 7.

[8] http://projects.semwebcentral.org/frs/?group_id=156
[9] http://projects.semwebcentral.org/prjects/owls2wsdl

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-core"
namespace { _"http://dmas.dfki.de/axis/services/BookPrice#",
            _"http://www.soa4all.eu/benchmark1#",
            dc _"http://purl.org/dc/elements/1.1#",
            foaf _"http://xmlns.com/foaf/0.1/",
            wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
            loc _"http://www.wsmo.org/ontologies/location#",
            oo _"http://example.org/ooMediator#",
            context _"http://servicedesign.org.uk/soa4all/ontology/context#"}
/**********************************************************************
* WEB SERVICE
**********************************************************************/
webService _"http://dmas.dfki.de/axis/services/BookPriceService"
    nonFunctionalProperties
        context#hasTypeOfUSe hasValue forBusiness
        context#hasPlatform hasValue HandheldDevice
        context#hasUsageFrequency hasValue Low
    endNonFunctionalProperties

importsOntology {_"http://127.0.0.1/ontology/books_ontology",
_"http://127.0.0.1/ontology/Mid-level-ontology", _"http://127.0.0.1/ontology/concept_ontology"}

capability _"http://dmas.dfki.de/axis/services/BookPriceCapability"
    sharedVariables {?typ, ?boo}

    //PreCondition
    precondition definedBy
        ?boo memberOf Book-Type .

    //PostCondition
    postcondition definedBy
        ??typ memberOf Price.
```

Fig. 7. BookPriceService.wsml attached with contextual information

5.2 Impact of our Semantic Content-based Approach

We compare our approach with state-of-the-art content-based approaches considering semantic similarities as presented in [Klusch and Kapahnke 2010]. The particular tool used is SME2 v2.2[10]. And iSeM v1.1[11] is used as the matchmaker plugin. Three adaptive hybrid approaches within the plugin are selected for the evaluation:

SVM1: the combination of SVM logic-based, text-similarity and structure as the first approach;

SVM2: the combination of SVM logic-based, text-similarity, structure, approximated logic-based, and specification as the second approach;

SVM3: the combination of SVM logic-based, text-similarity, structure, and specification.

Four services are selected to serve as prototypes of user queries from the test set (service1 is named as book_Price_service, service2 is named as bookpersoncreditcard-account_service, service3 is named as bookpersoncreditcardaccount_price_service, and service4 is called university_lecture_in_academia_service). Then we establish four ranking lists of services relevant to these user inputs for each of the four competing approaches. For each user request, the first 20 of the ranking results are then compared to its relevance set given by the test collection.

Classical precision and recall [Klusch et al. 2010], F1 are used as the performance evaluation:

$$Precision = \frac{|A \cap B|}{|B|}, Recall = \frac{|A \cap B|}{|A|}, F1 = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision} \quad (8)$$

[10] http://www.semwebcentral.org/projects/sme2/
[11] http://projects.semwebcentral.org/projects/isem

Where A is the set of all relevant documents, which are provided within the test collection, and B the set of all retrieved documents for a query.

The evaluation results are shown in Figure 8.

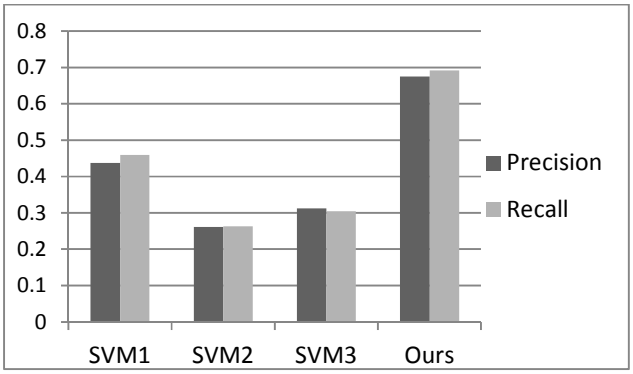

Fig. 8. Precision/Recall by iSeM and our approach

In our evaluation of the four randomly selected queries, SVM 1 (the combination of SVM logic-based, text-similarity and structure) performs better than SVM 2 and SVM 3. Our semantic content-based approach performs the best.

Table III. F1 by iSem and our approach

	SVM1	SVM2	SVM3	Ours
F1	0.4473	0.2615	0.3082	0.6815

5.3 Impact of Context Similarity Measure

In order to evaluate the effect of contextual information, services are associated with contextual information. For each user input, our approach provides 30 ranked services listed according to their semantic content similarity to the input. We have then asked six experts who have experience either in the development or the use of web services to rank the ten most relevant services to each input from the provided list. They have also been asked to give an explicit expression on the importance of context that plays in their selection decision in this short questionnaire gathering.

The impact of the context similarity measure is evaluated against experts' opinions in two ways. Firstly, we compare our approach which considers both context and content information, with the questionnaires results from experts. Secondly, we only compare our semantic content-based approach with the questionnaire results. The comparison of the two evaluations above can indicate effectiveness of the involvement of context similarity. By doing so, the performance of our approach can also be evaluated against the opinions of experts. The same precision, recall and F1 metrics as above are used for evaluation.

To compare our approach with the opinions of experts, we apply the average weight our experts attached to context in Equation (7) to create a new ranking list of our services against each of the four queries for the same target context values as shown to experts. The top most 10 highly ranked services have been selected as the recommendations provided for each service. However, the services with the same ranking score as the tenth service are also included in the retrieved set. The ten selected services by experts for each service are included in the relevant set. Since the numbers of services within shared set and retrieved set are very similar to each

other, we have very similar values in precision, recall or F1. The selected services by experts are also used to compare with the top ten services ranked by the semantic content-based approach, in order to evaluate the impact of contextual information.

Figure 9 presents the evaluation metric F1 of our approach against the opinions of experts. Generally, the F1 of recommendations reaches a reasonable level, which shows that the context similarity metric underpinning our approach is working. From the figure we can see that the metric F1 of service2 (bookpersoncreditcardaccount_service) and that of service3 (bookperson-creditcardaccount_price_service) are not as good as the other two services. We hypothesize that this is due to deficiency in the way in which information about service functionality is recorded within the service name, and the absence of more detailed and precise information. Thus, because service2 and service3 are described in a general way, they seem very similar to each other, while service1 and service4 are labeled with more detailed descriptions.

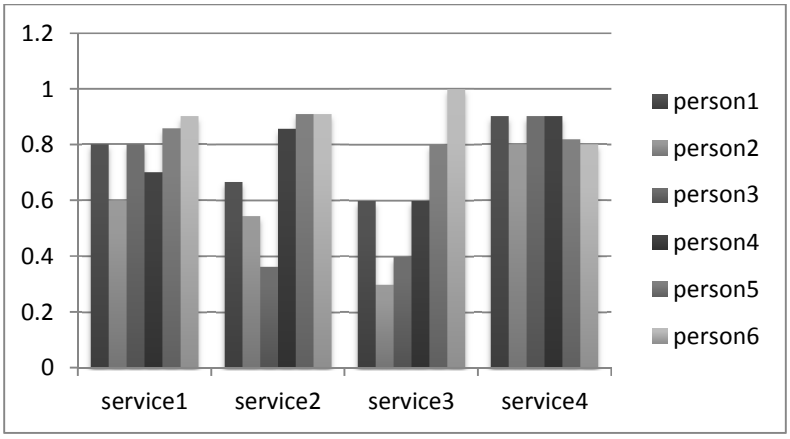

Fig. 9. F1 of context similarity measurement

The difference in the results of the comparison between semantic content-based approach which considers context and experts, and the comparison between semantic content-based approach which does not consider context and experts, indicates the impact of the contextual information plays in the recommendation.

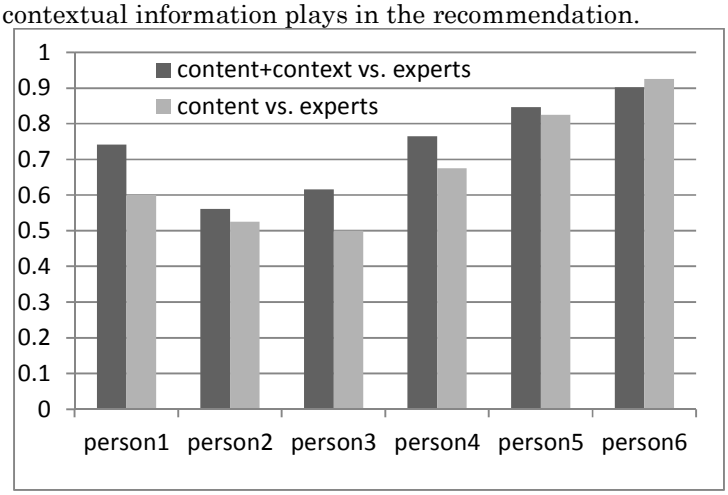

Fig. 10. The impact of involving contextual information evaluated by F1

From Figure 10, we can see the performance of the semantic content-based context approach is generally better than the semantic content-based approach in terms of F1. This figure also indicates the importance of including contextual information and the effectiveness of using our context similarity metric in recommendations.

6. CONCLUSION AND FUTURE WORK

This work presents an approach to effective recommendation of software services, combing context-aware analysis with semantic content-based recommendation. It has several advantages. Firstly, it addresses the dearth of existing feedback, common to new domains like web services through a semantic content-based approach. Secondly, the context analysis adapts the recommendation to the context in which a service is to be used so that a more precise list of results will be returned. The final recommendations are generated by combining the results from both semantic content-based approach and our context analysis. Our approach not only enhances the recommender systems with non-standard DL reasoning on service contents without limiting the prediction to the user's historical preferences, but also covers future potential extensions by considering contextual information. At the end, our semantic content approach is measured through an existing software service dataset, and its effectiveness and advantage over the three alternatives approaches are demonstrated. The context analysis has been evaluated through comparing our recommendations with users' selections. The early stage of this work has highlighted some deficiencies of our approach. For example in our approach, the service web portals or service providers may provide different context parameters for services, which make context analysis potentially time consuming.

Our further work will therefore be focused on building a software service website interface to collect users' feedback based on user's profiles and multiple contexts. By considering users' explicit feedback we can reduce the inherent weakness introduced by assuming that users like the service they have purchased. One of the other extensions will be considering user's profile or preference, such as which service areas, and the general requirements on services. This area of 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

Adomavicius, G., Sankaranarayanan, R., Sen, S. and Tuzhilin, A. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS).* 23, 1, 103-145.

Adomavicius, G. and Tuzhilin, A. 2005. 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, 6.

Anand, S. S. and Mobasher, B. 2005. Intelligent Techniques for Web Personalization. *Intelligent Techniques for Web Personalization*. Springer Berlin / Heidelberg. 3169/2005. 1-36.

Ankolenkar, A., Paolucci, M., Srinivasan, N. and Sycara, K. 2004. The Owl-s Coalition. *owl-s 1.1.*

Baader, F. and Nutt, W. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*.

Bennett, K., Layzell, P., Budgen, D., Brereton, P., Macaulay, L. and Munro, M. 2000. Service-based software: the future for flexible software. *Proceedings of Seventh Asia-Pacific Software Engineering Conference, APSEC 2000.* 214-221.

Bouquet, P., Kuper, G. M. and Zanobini, S. 2005. Asking and Answering Queries Semantically. *WOA*, 22-27.

Brandt, S., Küsters, R. and Turhan, A.-Y. 2002. Approximation and difference in description logics. *KR*, 203-214.

Brezillon, P. 2003. Focusing on Context in Human-Centered Computing. *IEEE Intelligent Systems*, 62-66.

Broens, T., Pokraev, S., Sinderen, M. v., Koolwaaij, J. and Costa, P. D. 2004. Context-aware, Ontology-based Service Discovery. *Lecture Notes in Computer Science*. Eds. Springer-Verlag Berlin Heidelberg. Volume 3295/2004, 72-83.

Bruijn, J. d., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Kifer, M., König-Ries, B., Kopecky, J., Lara, R., Oren, E., Polleres, A., Scicluna, J. and Stollberg, M. 2005. Web Service Modeling Ontology (WSMO). from http://www.wsmo.org/TR/d2/v1.2/20050413/.

Cohen, W. W., Borgida, A. and Hirsh, H. 1992. Computing Least Common Subsumers in Description Logics. *AAAI*, 754-760.

Cordì, V., Lombardi, P., Martelli, M. and Mascardi, V. 2005. An Ontology-based Similarity between Sets of Concepts. *WOA*, 16-21.

Debaty, P., Goddi, P. and Vorbau, A. 2005. Integrating the physical world with the web to enable context-enhanced mobile services. *Mobile Networks and Applications*. 10, 4, 385-394.

Dietze, S., Mrissa, M., Domingue, J. and Gugliotta, A. 2010. Context-Aware Semantic Web Service Discovery through Metric-based Situation Representations. *Enabling Context-Aware Web Services Methods, Architectures, and Technologies*. Eds. Sheng, Q. Z., J. Yu and S. Dustdar, Chapman & Hall / CRC Press.

Fensel, D., Kifer, M., Vruijn, J. d. and Domingue, j. 2005. Web Service Modeling Ontology submission.

Garcia-Molina, H., Koutrika, G. and Parameswaran, A. 2011. Virtual Extension Information Seeking: Convergence of Search, Recommendations, and Advertising. *Communications of the ACM*. 54, 121-130.

Goble, C. and Roure, D. D. 2002. The Grid: an application of the Semantic Web. *ACM SIGMOD*.

Horrocks, I. R. 1998. Using an Expressive Description Logic: FaCT or Fiction? *KR*, 636-649.

Kaufer, F. and Klusch, M. 2007. Performance of Hybrid WSML Service Matching with WSMO-MX: Preliminary Results. *In First International Joint Workshop SMR2 2007 on Service Matchmaking and Resource Retrieval in the Semantic Web at the 6th International Semantic Web Conference (ISWC 2007)*. 63-77.

Klusch, M. and Kapahnke, P. 2010. iSeM: Approximated Reasoning for Adaptive Hybrid Selection of Semantic Services. *Lecture Notes in Computer Science*. 6089/2010, 30-44.

Klusch, M., Kapahnke, P. and Zinnikus, I. 2010. Adaptive Hybrid Semantic Selection of SAWSDL Services with SAWSDL-MX2. *International Journal on Semantic Web and Information Systems (IJSWIS)*. 6, 4, 1-26.

Kocaballi, A. B. and Kocyigit, A. 2007. Granular best match algorithm for context-aware computing systems. *The Journal of Systems and Software*. 80, 2015-2024.

Küsters, R. 2001. *Non-Standard Inferences in Description Logics*. Springer.

Lécué, F. and Delteil, A. 2007. Making the Difference in Semantic Web Service Composition. *AAAI*, 1383-1388.

Li, L. and Horrocks, I. 2003. A Software Framework for Matchmaking Based on Semantic Web Technology. *WWW*, 331-339.

Liu, L., Lécué, F. and Mehandjiev, N. 2011. A Hybrid Approach to Recommending Semantic Software Services. *The 9th International Conference on Web Services (IEEE ICWS 2011)*.

Liu, L., Lécué, F., Mehandjiev, N. and Xu, L. 2010. Using Context Similarity for Service Recommendation. *Fourth IEEE International Conference on Semantic Computing*.

Maamar, Z., Benslimane, D. and Narendra, N. C. 2006. What Can Context Do for Web Services? *Communications of the ACM*. 49, 98-103.

Maamar, Z., Mostefaoui, S. K. and Mahmoud, Q. H. 2005. Context for Personalized Web Services. *Proceedings of the 38th Hawaii International Conference on System Sciences*.

Manikrao, U. S. and Prabhakar, T. V. 2005. Dynamic Selection of Web Services with Recommendation System. *Proceedings of the International Conference on Next Generation Web Services Practices*.

McIlraith, S. A., Son, T. C. and Zeng, H. 2001. Semantic Web Services. *IEEE Intelligent Systems*, 46-53.

Medjahed, B. and Atif, Y. 2007. Context-based Matching for Web Service Composition. *Distrib Parallel Databases*. 21, 5-37.

Navarro, G. 2001. A Guided Tour to Approximate String Matching. *ACM Computing Surveys* 33, 1, 31-88.

Noia, T. D., Sciascio, E. D., Donini, F. M. and Mongiello, M. 2003. A System for Principled Matchmaking in an Electronic Marketplace. *WWW*, 321-330.

Paolucci, M., Kawamura, T., Payne, T. and Sycara, K. 2002. Semantic Matching of Web Services Capabilities. *ISWC*, 333-347.

Papazoglou, M. P. 2008. *Web Services: Principles and Technology*. Pearson Education Limited.

Pashtan, A., Kollipara, S. and Pearce, M. 2003. Adapting Content for Wireless Web Services. *IEEE Computer Society*, 79-85.

Sampson, S. E. and Froehle, C. M. 2006. Foundations and Implications of a Proposed Unified Services Theory. *Production and Operations Management*. 15, 2, 329-343.

Schafer, J. B., Konstan, J. A. and Riedl, J. 1999. Recommender Systems in E-Commerce. *Proceedings of the 1st ACM conference on Electronic Commerce*. 158 - 166

Schafer, J. B., Konstan, J. A. and Riedl, J. 2001. E-Commerce Recommendation Applications *Data Mining and Knowledge Discovery*. 5, 115-153.

Sivashanmugam, k., Verma, k., Sheth, A. and Miller, J. 2003. Adding Semantics to Web Services Standards. *ICWS*, 395-401.

Sreenath, R. M. and Singh, M. P. 2004. Agent-based service selection. *Journal of Web Semantics*. 261-279.

Terziyan, V. and Kononenko, O. 2003. Semantic Web Enabled Web Services: State-of-Art and Industrial Challenges. *Web Services - ICWS-Europe 2003*. Springer Berlin/Heidelberg. 2853/2003, 183-197.

Zheng, Z., Ma, H., R.Lyu, M. and King, I. 2009. WSRec: A Collaborative Filtering Based Web Service Recommender System. *IEEE International Conference on Web Services*. 437-444.

Received February 2012; revised Oct 2012; accepted Mar 2013