Winter 12-2-2007

E-Business Service Semantic Classification and Retrieval

Aviv Segev

Eran Toch

Proceedings of The Seventh International Conference on Electronic Business, Taipei, Taiwan, December 2-6, 2007, pp. 277-289.

E-BUSINESS SERVICE SEMANTIC CLASSIFICATION AND RETRIEVAL

Aviv Segev, National Chengchi University, Taiwan, asegev@nccu.edu.tw
Eran Toch, Technion - Israel Institute of Technology, Israel, erant@tx.technion.ac.il

ABSTRACT

In this work we provide an initial analysis of e-business service classification and retrieval. We present a process of associating e-business services with ontologies and show how the process can be similarly used for e-business service retrieval. The semantic understanding of business services may provide added value through the creation of new compositions of services. We analyze two common methods for text processing, TF/IDF and context analysis. We also compare the use of service description as a means for free text retrieval of categorized WSDL description of web service. Our initial results indicate that context analysis is more useful than TF/IDF and that free textual descriptions can be used for retrieval of categorized WSDL description using similar processes.

Keywords: e-service, classification, retrieval, semantic

INTRODUCTION

Electronic businesses today are increasingly developing new systems and services based on existing components. Services based on extant software resources supply immediate value creation to an organization. An electronic business can utilize the advantages of using existing Web services to build new applications. Standards for Web services, such as WSDL, are already available. However, the problem of identifying and integrating e-Services is a challenging task. We present semantic methods to enable businesses to classify existing modules to facilitate their integration into new services: the process of associating services with ontologies. Service classification is an important pre-processing step for tasks such as service composition and rapid e-business construction.

Services are autonomous code packages, which can be utilized by businesses to provide e-Services through the Internet. These units were independently developed and hence are characterized by different platforms, programming languages, and interfaces. This lack of homogeneous structure requires a method that defines the automatic communication between services so as to enable the integration between them.

However, as communications protocols and message formats become standardized in the web community, it is increasingly possible and important to be able to describe the communications in some structured manner. For example, WSDL addresses this need for standardization by defining a grammar that describes network services as collections of communication endpoints able to exchange messages. WSDL service definitions supply documentation for distributed systems and provide guidelines for automating the process of applications communication.

Previous methods based the process of matching services on annotated textual description. Works on service integration focused on various aspects of the problem. The issue of interface heterogeneity was addressed by the use of semantic web services. Using languages such as OWL-S [1], the scope of Web services is extended with an unambiguous description by relating properties such as input and output parameters to common concepts and by defining the performance characteristics of the service. The concepts are defined in *Web ontologies* [2], which serve as the key mechanism to globally define and reference concepts. Service composition through planning (in the AI sense) was introduced [12, 3] to manage the complexity of the problem.

In [21] and [20], it is argued that the process of service composition may have an **exploratory** nature rather than one of planning. Therefore, it is often the case that only partial solutions to composer requirements exist, as Web services are created autonomously without any a-priori knowledge of their intended use.

Furthermore, composer requirements may not be well-defined. Rather, they may be driven by the availability of Web services. This type of usage requires iterative composition and selection of partial services, rather than the design of the complete composition. In an attempt to support exploratory composition, an

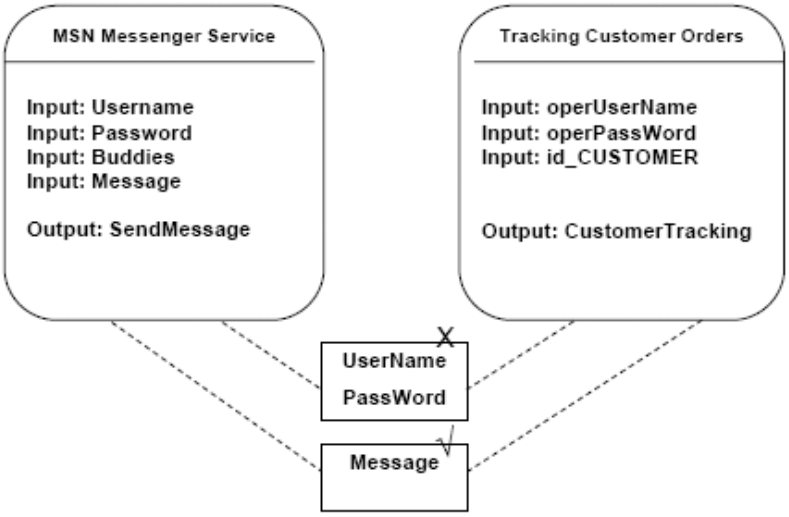

Fig. 1. Unite service example

engineering approach was taken to provide *approximate service retrieval*, in which the best effort composition is followed by "gluing" services together using some additional programming work.

Since these methods do not address semantical meaning, the matching of services (*e.g.*, for the purpose of composition) may provide semantically incorrect results, especially in the creation of new business services that seek to integrate existing modules. This problem was addressed by [6]. They presented a general method of associating semantic meaning with services in an ontology so as to facilitate better matching.

This paper focuses on the critical element of semantical meaning that promotes better matching. We present electronic businesses with a semantics-based method that allows the identification of existing services that can be integrated to create new applications for the organization's benefit. By identifying the existing services that can be integrated, we enable the more rapid and more complete construction of new business applications.

Take, for example, two real-world Web services, illustrated in Figure 1. The two services, *MSN Messenger Service* and *Tracking Customer Orders*, share some common concepts, such as the *UserName* and the *PassWord* concepts. These two services originate from very different domains. The first is concerned with business and the second with communication. These two services might be considered for a meaningful composition: a business might be interested in tracking customer orders using the MSN messenger service. However, this example demonstrates that methods based exclusively on mapping the concepts to the service's parameters (such as in [14]) may yield inaccurate results. In our example, the *UserName* and the *PassWord* concepts represent totally different organizational points of view. In the *Tracking Customer Orders* the username represents the system operator who is tracking the client. Conversely, in the *MSN Messenger Service* the username and password represent the calling user. We expect that the two services will share the *Message* that is transferred between them, the output from the one service acting as the input to the other. However, only one service carries this concept and thus we seek to link between the services and offer businesses the possibility of combining the two services.

Therefore, in this work we propose to implement in the field of business the use of *service classification* and *service retrieval*. Service classification defines a process that classifies a service to a set of concepts (or an ontology) that represents a domain. Service retrieval represents a similar process that can extract relevant business services based on free textual description. The classification and retrieval of services to and from their respective domains can be used to classify the validity of the service composition and to rule out compositions of unrelated services. We present methods that spotlight possible compositions of innovative business services using existing ones.

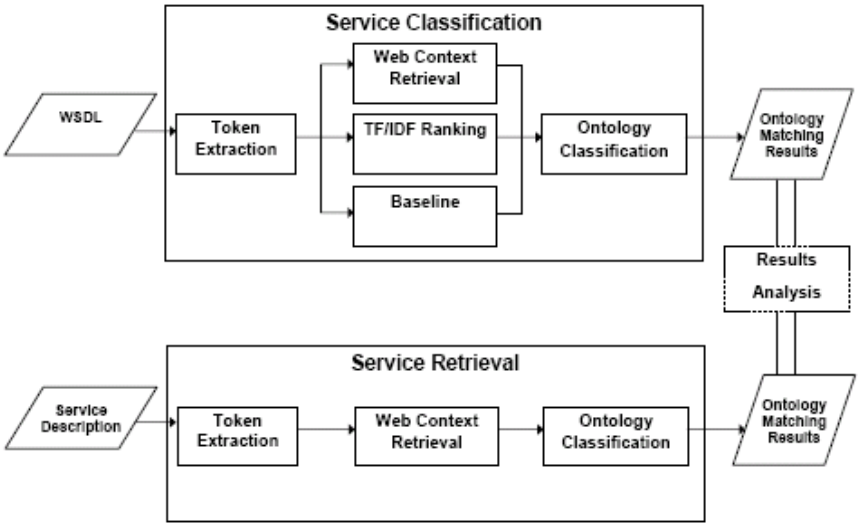

Fig. 2. The similar business service classification and business service retrieval processes

We provide a preliminary study, showing how e-business service semantic classification and retrieval can be performed based on the use of two known methods, TF/IDF [18] and context generation [19]. We propose a three dimensional classification of methods for associating a Web service with an ontology concept and provide a similar technique for the retrieval of Web services. We present partial experiments to test various combinations of these known methods for the classification and retrieval, using a real-world data set. Our analysis indicates that context analysis is more useful than TF/IDF for WSDL classification and that context analysis can be used for free textual description e-service retrieval.

The rest of the paper is organized as follows. Next we provide a simple model for service classification and service retrieval using the three dimensional classification. In following section we present the design and results of the experimental evaluation of our work. The next section describes related work. Finally, the last section concludes the research and provides directions for future work.

BUSINESS SERVICE CLASSIFICATION AND RETRIEVAL

Overview

In this section, we describe how the business process can utilize the model described in [6] to categorize new business services and to search for existing e-business services. We examine each business process from two points of view. One point of view is based on the syntactic properties of the service that will be associated with an ontology concept. The other point of view is the textual description of a service which can also be associated with an ontology concept. Figure 2 details the stages of the e-service classification on the top process flow. The bottom business process flow portrays the textual service retrieval stages. For the classification process, different evaluation methods are used. Since web services usually appear with both a WSDL document describing the syntactic properties of the service interface and a textual description, we can separate these two descriptions and treat them as distinct inputs. We can then analyze and compare the results - the ontology concepts to which each of the two processes maps.

Figure 3 depicts an example of these two descriptions. The top describes the syntactic properties of the TrackingAll e-business service. The bottom includes the textual description that accompanies this service. The spelling mistake in the original textual description emphasizes the difficulties associated with the business service classification and retrieval process.

Three methods were used for the analysis of the e-business service classification: TF/IDF, Web context extraction, and a *baseline* for evaluation purposes. The Web context extraction method was also used for the

business service retrieval. The baseline method is a simple reflection of the original bag of tokens extracted from the service descriptions. The basic data structure used by all the methods is a ranked bag of tokens, which is processed and updated in the different stages. After the different analysis methods were applied, the final classification and retrieval are obtained by matching the bag of tokens to the concept names, attributes, and documentation of each of the ontologies.

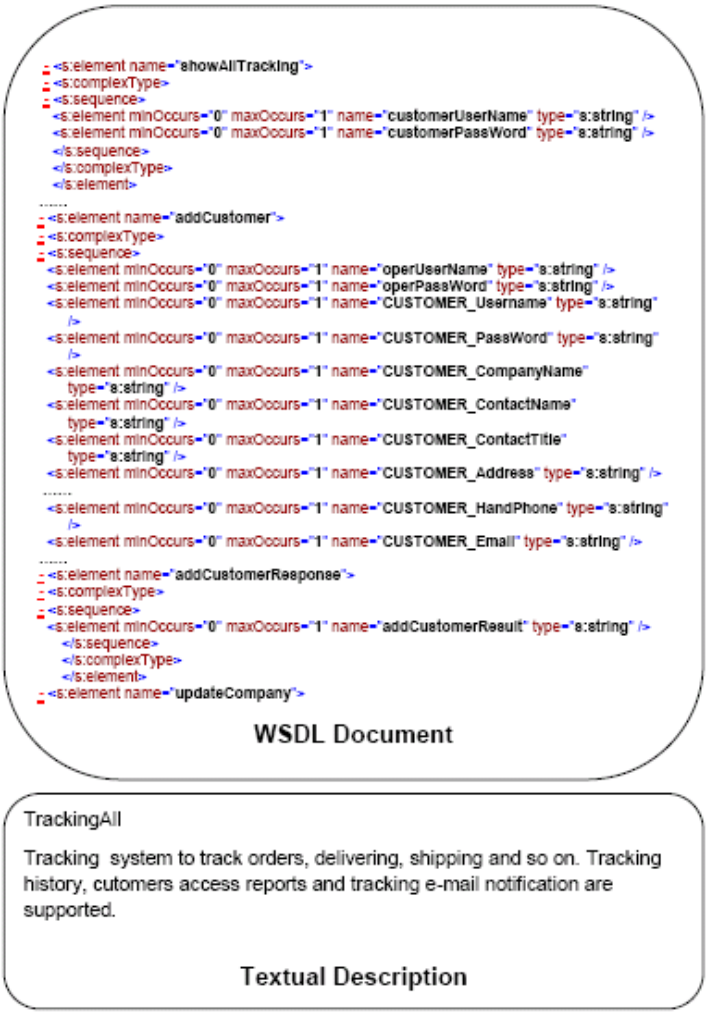

Fig. 3. Example of the tracking customer orders service

Service Classification Analysis

The service analysis is based on token extraction, representing each service, S, using sets of tokens, called *descriptors*. Each token is a textual term, extracted by simple parsing of the underlying documentation of the service. The descriptor represents the WSDL document, formally put as $D^S_{wsdl} = \{t_1, t_2, \ldots\}$. WSDL tokens require special handling, since meaningful tokens (such as parameter names and operation names) are usually composed of a sequence of words, with the first letter of each word capitalized (*e.g.*, setCustomerPermission) or separated by an underscore (*e.g.*, CUSTOMER_Username). Therefore, the tokens are divided into separate tokens. The tokens are filtered using a list of *stop-words*, removing words with no substantive semantics. For instance, the tokens *get*, *response*, and *result* are common in many WSDL documents.

An illustration of the baseline token list is depicted in Figure 4. These tokens were extracted from the WSDL document. All elements classified as name were extracted. The sequence of words was expanded as previously mentioned using the first capital letter of each word or underscore separating the words.

Service Retrieval Analysis

We analyze the service retrieval based on the textual descriptions accompanying each of the services. These textual descriptions vary in length from a sentence to a paragraph describing the service. The textual descriptions are usually released with the e-business service. Since these free text descriptions are written by the service developing party we assume they most accurately define the service concerned.

The service retrieval analysis is based on similar token extraction, representing each service query, S, and using the *descriptors*, sets of tokens. Each token is a textual word from the query, extracted by parsing the documentation of the service according to space separators. The query service descriptor, $D^S_{query} = \{t_1, t_2, \ldots\}$, represents the textual description of the service.

An example of the query descriptor tokens would be all of the words appearing in Figure 3 bottom. The original spelling of the description, including the spelling mistakes, was kept in order to represent similar query conditions.

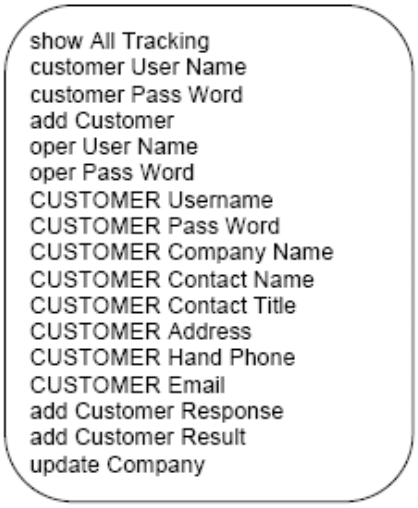

Fig. 4. An example of the baseline representation of the customer order service

Analysis Methods

In this section we describe the two methods used for the classification and retrieval of the e-business services, TF/IDF and context extraction, and present the motivation for choosing these two methods. Nevertheless, this choice is more or less arbitrary. Other methods for text extraction from the vast literature of Information Retrieval (IR) [16] and Machine Learning (ML) [13] can be used.

TF/IDF Analysis TF/IDF (Term Frequency / Inverse Document Frequency) is a common mechanism in IR to create a robust set of representative keywords from a corpus of documents. The method can be applied here separately to the WSDL descriptors and the textual descriptors since the linguistic characteristics of the

two document types are very different. By building an independent corpus for each document type, irrelevant terms are more distinct and can be eliminated with a higher confidence. In order to formally define TF/IDF, we start by defining $freq(t_i, D_i)$ as the number of occurrences of the token t_i within the document descriptor D_i. We define the term frequency of each term as:

$$tf(t_i) = \frac{freq(t_i, D_i)}{|D_i|}$$

We define $\mathcal{D}_{wsdl}$ to be the corpus of WSDL descriptors and $\mathcal{D}_{query}$ to be the corpus of textual descriptions. The inverse document frequency is calculated as the ratio between the total number of documents and the number of documents that contain the term:

$$idf(t_i) = \log \frac{|\mathcal{D}|}{|\{D_i \,:\, t_i \in D_i\}|}$$

Here, $\mathcal{D}$ is defined generically, and its actual instantiation is chosen according to the origin of the descriptor. Finally, the TF/IDF weight of a token, annotated as $w(t_i)$ is calculated as:

$$w(t_i) = tf(t_i) \times idf^2(t_i)$$

While the common implementation of TF/IDF gives equal weights to the term frequency and inverse document frequency (*i.e.*, $w = tf \times idf$), we have chosen to give higher weight to the *IDF* value. The reason behind this modification is to normalize the inherent bias of the *TF* measure in short documents [17]. While traditional TF/IDF applications were concerned with verbose documents (such as books, articles and human-readable Web pages), WSDL documents and the textual description of services are relatively short. Therefore, the frequency of a word within a document tends to be incidental, and the document length component of the TF generally has no impact.

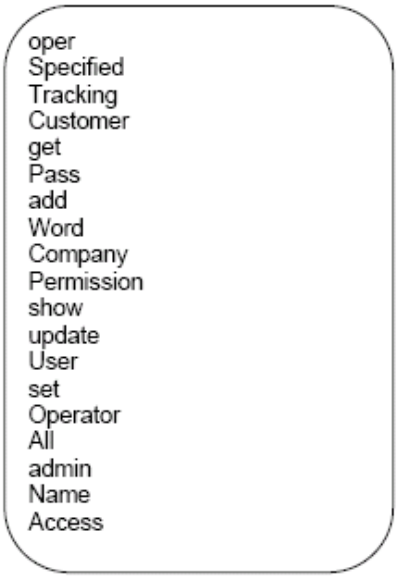

Fig. 5. An example of the TF/IDF high scored list of the customer tracking service

The token weight is used to induce ranking over the descriptor's tokens. We define the ranking using a precedence relation $\preceq_{tf/idf}$, which is a partial order over D, such that $t_l \preceq_{tf/idf} t_k$ if $w(t_l) < w(t_k)$. The ranking is used to filter the tokens according to a threshold which filters out words with a frequency count higher than the second standard deviation from the average frequency. The effectiveness of the threshold was validated by our experiments. Figure 5 presents the list of tokens which received a higher weight than the

threshold. Several tokens which appeared in the baseline list (see Figure 4) were removed due to the filtering process. For instance, words such as "Response" and "Result" received below-the-threshold TF/IDF weight, due to their high frequency.

Context Extraction The extraction process uses the World Wide Web as a knowledge base to extract multiple contexts for the tokens. Extraction is used to filter out biased tokens, to provide a more precise ranking, and to extend the service descriptors. The algorithm input is defined as a set of textual propositions representing the service description. The result of the algorithm is a set of *contexts* - terms that are related to the propositions. The context recognition algorithm was adapted from [19] and consists of the following three steps:

1. **Context retrieval:** Submitting each token to a Web-based search engine. The contexts are extracted and clustered from the results.
2. **Context ranking:** Ranking the results according to the number of references to the keyword, the number of Web sites that refer to the keyword, and the ranking of the Web sites.
3. **Context selection:** Finally, selecting the set of contexts for the textual proposition, defined as the *outer context*, $\mathcal{C}$.

The algorithm can formally be defined as follows: Let $\mathcal{D} = \{P_1, P_2, ..., P_m\}$ be a set of textual propositions representing a document, where for all P_i there exists a collection of descriptor sets forming the context $\mathcal{C}_i = \{\langle c_{i1}, w_{i1}\rangle, ..., \langle c_{in}, w_{in}\rangle\}$ so that $ist(\mathcal{C}_i, P_i)$ is satisfied. In our case the adopted algorithm uses the corpus of WSDL descriptors, $\mathcal{D}_{wsdl}$, as propositions P_i, and the contexts describing the WSDL as tokens c_i with their associated weight w_{i1}. McCarthy [11] defines a relation $ist(\mathcal{C}, P)$, asserting that a proposition P is true in a context $\mathcal{C}$. The context recognition algorithm identifies the outer context $\mathcal{C}$ defined by:

$$ist(\mathcal{C}, \bigcap_{i=1}^{m} ist(\mathcal{C}_i, P_i)).$$

The input to the algorithm is a stream, in text format, of information. The context recognition algorithm output is a set of contexts that attempts to describe the current scenario most accurately. The algorithm attempts to reach results similar to those achieved by a human when determining the set of contexts that describe the current scenario (the Web service in our case). For example, Figure 6 provides the outcome of the Web context extraction.

One of the most interesting properties of the Web context extraction analysis is its ability to add new, relevant, words. For example, the algorithm removed the word "tracking," which appeared in the baseline and the TF/IDF token lists, and introduced the words "marketing" and "shipping" which are more relevant to the business domain (for which the service actually belongs). This is an example of the advantages the Web context extraction approach has over the TF/IDF and baseline approaches.

Ontology Matching In this final step the semantically extracted token set can be treated similarly in both the service classification and the service retrieval processes. This step matches the finalized semantically extracted token set with the ontological concepts. Let $O_1, O_2, \ldots, O_n$ be a set of ontologies, each representing different domain knowledge. We provide a simplified representation of an ontology as $O \equiv \langle C, R\rangle$, where $C = \{c_1, c_2, \ldots, c_n\}$ is a set of concepts with their associated relation R.

In order to evaluate the matching of the concepts with the service descriptor, we use a simple string-matching function, denoted by $match_{str}$, which returns 1 if two strings match and 0 otherwise. We define S as the service, and recall that D^S is the service descriptor. Also, we define n to be the size of D^S. The overall match between the ontology and the service is defined as a normalized sum of the concept matching values:

$$match(S, O_i) = \frac{1}{n} \sum_{c_j \in O_i} \sum_{t_i \in D^S} match_{str}(t_i, c_j)$$

To conclude our example, in the baseline and the TF/IDF analysis, the two services mentioned in Section 1, the MSN Messenger Service and the Tracking Customer Orders process, were mapped to the same ontology

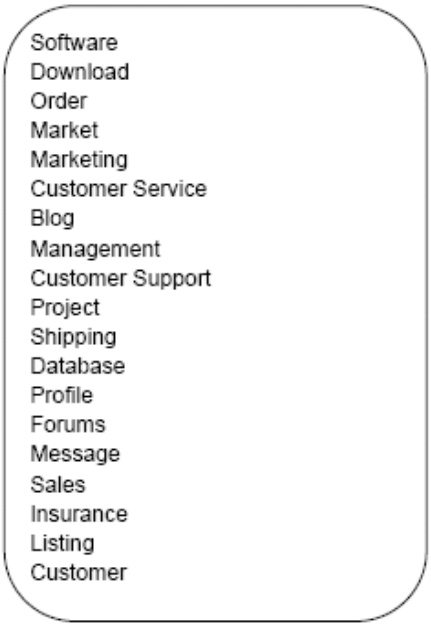

Fig. 6. An example of the web context

based on User and Pass, which are misleading descriptors in this case since one relates to the *business* ontology and the other to the *communication* ontology. Using context analysis, they were matched together not only to the same ontology but also under the same ontology concept - *Message*.

Discussion

The e-business service classification described in the service description supplied by the service developer and service provider is inefficient in specifying the classification, due to the perspective of the developer and provider and the terms they use. Another problem is that the developer is not aware of all the existing ontologies and all their concepts when providing a service. Furthermore, the provider cannot be forced to supply a detailed description. These textual descriptions usually consist of a bare minimum of information, which sometimes does not add to the understanding of the service.

The technique of e-business classification and retrieval proposed here can enable companies and applications to quickly, easily, and dynamically find and use Web services over the Internet, thus promoting the better exploitation of existing web services. This technique comprehensively addresses the problems of web service development through the re-use of existing autonomous code packages.

An important feature is that the model is not industry-specific. The model can be implemented in any field due to its ability to extract tokens based on knowledge retrieved from the Internet. Any industry, worldwide, offering products and services can benefit.

The model makes it possible for businesses to quickly identify web services which can be composed. Using the model of web service classification and retrieval new services can be created based on existing classified services. Additionally the model can suggest to the business solution developer possibilities that previously had not been thought of.

This model has a number of immediate benefits for businesses. The use of existing web services to compose new services enables the conservation of valuable organizational resources and promotes the speed of development of these new services. It allows businesses to invoke existing Web services to comprise new ones - so as to provide added value to their customers.

We conclude this section with a worst case performance analysis. The complexity analysis of the TF/IDF method yields $o(mn)$, where m is the number of WSDL documents and n is the number of tokens. The complexity of the context Web-based method is $o(an)$, where n represents the number of input cycles, such

as each line of text. The a represents a constant limiting of the number of top ranking results from each cycle of the algorithm. The context method performance execution time is higher than the TF/IDF, since it needs to access the Web search engine for every line of input extracted from the WSDL, and can reach between 3 to 4 minutes for very long WSDL documents. However, since each web service only needs to be classified once in its lifetime, performance is less crucial than accuracy. For the retrieval process, the performance is much faster and is up to 5 seconds for a query that is composed of a few sentences.

EMPIRICAL ANALYSIS

In this section we describe our experiments and provide some empirical analysis and comparison of the different business service classification and service retrieval methods.

Experimental Setup

The data for the experiments were taken from an existing benchmark repository, of several hundred Web services, provided by the researchers from University College Dublin.[1] Our preliminary experiments use a set of 29 representative Web services, divided into 4 different topics: courier services, currency conversion, communication, and business. For each Web service the repository provided a WSDL document and a short textual description. The ontologies that were used for the comparison were taken from another repository, named OWLS-TC [9], which includes over 40 different ontologies from various domains.

The experiments examined three different methods for business service classification and one method for business service retrieval, as described in Section 1. The service classification methods included:

Name Context The Context Extraction algorithm described in Section 1 was applied to the *name* labels of each Web service. Each descriptor of the Web service context was used as a token.

Name TF/IDF Each word in the document was checked for term frequency and inverse document frequency (TF/IDF). The set of highest ranking weighted value words was used.

Our experiments compared the two methods, with an addition of a **baseline**, which included the original token list extracted from the service descriptors. The actual comparison was based on mapping the output of each of the methods to the set of ontologies, using the string matching method described in Section 1.

The business service retrieval was based on the textual description of each service. The Context Extraction algorithm was applied to the textual description of the Web services forming the **Descriptor Context**. Each descriptor of the Web service context was used as a token.

Performance Measure The metrics in our experiments were recall and precision. Recall that services can be classified into several domains and therefore we show, via example, how these measures were computed. Assume that a given service is judged by a human observer to belong to ontologies A, B, and C. Now, assume that a method classifies service X with ontologies A, B, D, and E. We penalize this method for not choosing C and for choosing D and E. Therefore, the precision should be 2/4 since only 2 out of the 4 ontologies we provide are a match. Recall should be 2/3, since we managed to identify 2 out of the 3 ontologies to which the service belongs.

Experimental Results

In our first experiment we analyze the usefulness of going beyond the baseline bag of tokens. Figure 7 compares the precision and recall of the methods of classification and retrieval. The baseline method achieved 100% recall but only 11% precision. This result means that the baseline has sufficient tokens to match with almost all of the ontologies for each service. Clearly, this phenomenon shows poor selectivity, as shown by the low precision level. The TF/IDF improved the precision to 17% while keeping the recall at 100% due to elimination of some of the most general tokens which belong to most ontologies. The best result, dominating all others, was achieved by the Name Context method, yielding 100% recall and precision of 37%. We can therefore conclude that the use of context generation has significant impact on the success of text classification. Further improvement is needed to increase precision even more.

[1] http://moguntia.ucd.ie/repository/ws2003.html

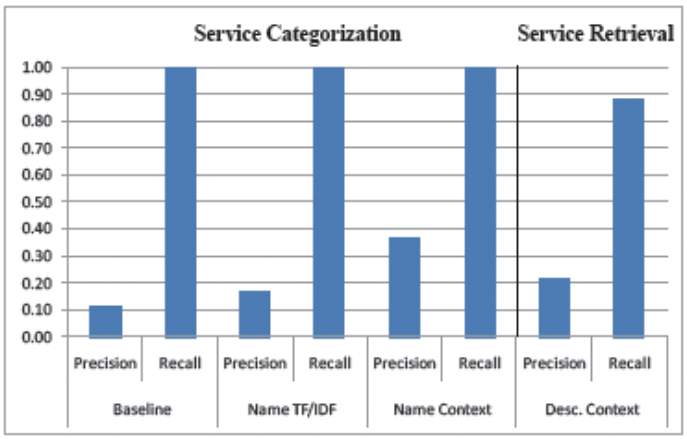

Fig. 7. Precision and recall of all methods

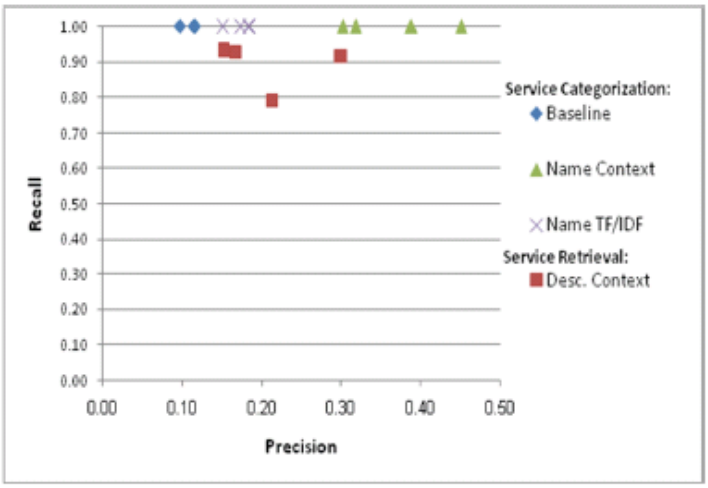

Fig. 8. Precision vs. recall of service classification and service retrieval

Data Regarding the service retrieval method, the Description Context managed to increase the precision to 22%. However, the cost was an 11% decrease in recall.

Our aim is to improve the precision while maintaining the level of recall, and thus the integration of the Name Context and the Name TF/IDF methods should be considered. Since these two methods work differently to extract tokens, the integration of the Name Context method with the Name TF/IDF method should boost the precision through the examination of the overlap between the two resultant sets of ontologies.

Precision can be improved by pre-processing the ontologies themselves. TF/IDF can be applied to filter out common concepts by using the corpus of ontologies. Thus, generic concepts, such as *market*, can be replaced with more precise concepts, such as *stock-market* and *fish-market*. Classification can be improved, by relying only on truly identifying tokens.

Figure 8 displays the precision vs. recall of the classification and retrieval, partitioned into topics. The results are presented on a precision/recall graph, where precision is given on the x-axis and recall on the y-axis. We can see that the performance of the Baseline method for all topics is dominated by the Name TF/IDF and Name Context methods. The figure shows all four baseline results, yet they overlap in pairs. The results of the TF/IDF method are also clustered together, at a slightly higher value. The Name Context

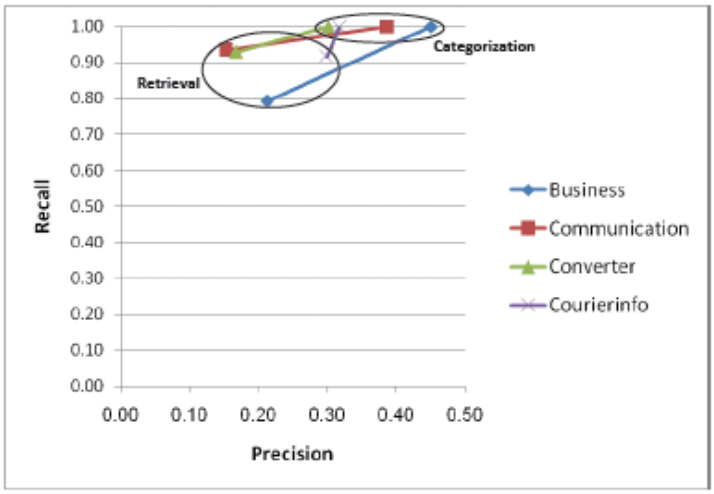

Fig. 9. Precision vs. recall of classification and retrieval according to each topic

method dominates all other methods, achieving the highest precision and recall levels for all topics. For the Business domain, the method achieved precision of 45% and recall of 100%.

The results of the service retrieval using the Description Context method are more dispersed, with a lower recall than all the other methods and a precision that overlaps both with the TF/IDF method and with the lower precision values of the Name Context method.

The results of the classification and retrieval can be combined using different metrics. For example, the classification and retrieval can be aggregated using probability multiplication, measures averaging, finding minimum and maximum, etc.

Figure 9 displays the precision and recall for the best performing classification method, Name Context, and the retrieval results, using the Description Context method, achieved according to each topic. The purpose of this figure is to examine whether there is a considerable difference between the classification and retrieval in certain topics.

Figure 9 shows that in the topic of *Business* there exists the largest difference between classification and retrieval results of recall and precision 0.32, followed by the topics of *Communication* 0.24 and *Converter* 0.15. The lowest difference between the results was achieved by the *Courier Services*.

However, in the case of *Courier Services* we assume that the small difference between the classification and retrieval methods is due to low classification success of this specific method with the topic. This can be attributed to the *Courier Services* being a smaller domain with more tokens appearing in the labels. In the topics of *Business* and *Communication*, which are broader in scope, the name labels do not necessarily specify the topic and hence only the name context method was successful in inferring the right tokens.

RELATED WORK

Dong et al. proposed a search engine for Web services, Woogle [4]. It accepts keyword queries and returns results according to information in WSDL documents, such as message parameters. Although some of the matching algorithms used by Woogle are relevant to our work, Woogle matches keywords and our work explores the matching of formal concepts. We were able to provide further empirical evidence for some of the conclusions of Dong et al.: the effectiveness of clustering tokens according to their mutual distance in the WSDL file.

ASSAM [7] is a tool for semi-automatic annotation of Web services. It uses learning techniques to narrow down possible concepts, helping a human user manually tag the service. The objective of our approach is to provide a fully automatic labeling method (which is a coarser-grain task) and to classify and retrieve the service rather than to label service parameters.

Patil et al. [15] present a combined approach towards automatic semantic annotation of services. The approach relies on several matchers (string matcher, structural matcher, and synonym finder), which are

combined using a simple aggregation function. Duo et al. [5] present a similar method, which also aggregates results from several matchers. Our research aims at a coarser-grain task and we therefore chose different methods for our preliminary evaluation. However, we intend to evaluate the methods suggested in these works in future research.

Other models, such as [10] and [8], can be adopted for text classification as well. We choose to compare a set of three classification models and a retrieval model as a proof of concept.

CONCLUSION

In today's world, businesses need to adapt their services to the frequent changes in their business environment. Semantic classification and retrieval of services can assist businesses in their continuous struggle to improve their services and keep ahead of competitors by reducing the costs of generating new web services.

The ability to compose Web services based on free text and flexible composition can simplify the implementation of organizational needs. Our approach extends the scope of Web service utilization, by providing businesses with usable methods to investigate and access large scale service repositories. Rather than asking businesses to manually annotate their services with formal concepts, our method employs the information contained in the World Wide Web to establish rich context for user queries. Thus, we present a possible solution for the inherently poor description of Web services.

Our experiments prove, to some extent, the inherent problems of analyzing WSDL documents. Their short length and limited vocabulary cause serious challenges for labeling and classifying services. The weak performance of the TF/IDF measure, which works successfully on more verbose texts, proves that relying on the service text alone will not yield sufficient results.

We have described so far a work-in-progress. We intend to extend our experiments to a larger corpus of business services. Also, we encountered some problems with the use of general-purpose ontologies. Clearly, classification will work better with smaller, focused ontologies, and we intend to seek many such ontologies for a more rigorous experimentation. We will also look at methods for identifying coherent sub-ontologies based on classification results.

REFERENCES

[1] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. Daml-s: Semantic markup for web services. In *Proceedings of the International Semantic Web Workshop (SWWS)*, pages 411–430, July 13 2001.

[2] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein. OWL web ontology language reference. W3C candidate recommendation, W3C, 2004.

[3] J.G. Cardoso and A.G. Sheth. Semantic E-Workflow Composition. *Journal of Intelligent Information Systems*, 21(3):191–225, 2003.

[4] X. Dong, A.Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Simlarity search for web services. pages 372–383, 2004.

[5] Z. Duo, L. Juan-Zi, and X. Bin. Web service annotation using ontology mapping. In *SOSE '05: Proceedings of the IEEE International Workshop*, pages 243–250, Washington, DC, USA, 2005. IEEE Computer Society.

[6] A. Gal, A. Segev, and E. Toch. Semantic methods for service categorization - an empirical study. In *Proceedings International Workshop on Semantic Data and Service Integration (SDSI 2007)*, 2007.

[7] A. Heß, E. Johnston, and N. Kushmerick. ASSAM: A tool for semi-automatically annotating semantic web services. In *International Semantic Web Conference*, pages 320–334, 2004.

[8] A. Hotho, S. Staab, and A. Maedche. Ontology-based text clustering. In *Proceedings of the IJCAI-2001 Workshop Text Learning: Beyond Supervision*, 2001.

[9] M. Klusch, B. Fries, M. Khalid, and K. Sycara. Owls-mx: Hybrid semantic web service retrieval. In *Proceedings of 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web*. AAAI Press, 2005.

[10] T. Liu, Z. Chen, B. Zhang, W.-Y. Ma, and G. Wu. Improving text classification using local latent semantic indexing. In *ICDM*, pages 162–169, 2004.

[11] J. McCarthy. Notes on formalizing context. *In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993.

[12] B. Medjahed, A. Bouguettaya, and A.K. Elmagarmid. Composing web services on the semantic web. *VLDB J.*, 12(4):333–351, 2003.

[13] T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[14] M. Paolucci, T. Kawamura, T.R. Payne, and K.P. Sycara. Semantic matching of web services capabilities. In *International Semantic Web Conference*, pages 333–347, 2002.

[15] A.A. Patil, S.A. Oundhakar, A.P. Sheth, and K. Verma. Meteor-s web service annotation framework. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 553–562, New York, NY, USA, 2004. ACM Press.

[16] C. J. Van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979.

[17] S. Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520, 2004.

[18] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[19] A. Segev. Identifying the multiple contexts of a situation. In *Proceedings of IJCAI-Workshop Modeling and Retrieval of Context (MRC2005)*, 2005.

[20] E. Toch, A. Gal, I. Reinhartz-Berger, and D. Dori. A semantic approach to approximate service retrieval. *ACM Transactions on Internet Technology (TOIT)*, 8(1), 2008. forthcoming.

[21] E. Toch, I. Reinhartz-Berger, A. Gal, and D. Dori. OPOSSUM: Bridging the gap between web services and the semantic web. In Opher Etzion, Tsvi Kuflik, and Amihai Motro, editors, *NGITS*, volume 4032 of *Lecture Notes in Computer Science*, pages 357–358. Springer, 2006.