Semantic Methods for Service Categorization

An Empirical Study

Avigdor Gal, Aviv Segev, Eran Toch
Technion
Haifa
32000, Israel

ABSTRACT

In this work we provide an initial analysis of service categorization, the process of associating services with ontologies. Service categorization is an important pre-processing step to tasks such as service composition. The absence of semantic understanding of services may yield erroneous compositions and therefore, service categorization can assist in determining the correctness of a composition. We analyze two common methods for text processing, TF/IDF and context analysis. We also test two types of service description, textual and WSDL. Our initial results indicate that context analysis is more useful than TF/IDF and that WSDL description provides better categorization than textual description.

1. INTRODUCTION

In recent years, the use of services to compose new applications from existing modules has gained momentum. In part, this is due to the availability of simple standards such as WSDL for Web services. Also, the drive to utilize better existing resources (such as code reuse in organizations [6]) has driven organizations to seek technologies to maintain code repositories and code composition. Service composition in a distributed heterogeneous environment immediately raises issues of integration. Services are autonomous units of code, independently developed and evolved and therefore lack homogeneous structure beyond that of its interface (*e.g.*, as described in WSDL). Even there, while the interface may be a common one in the sense that all services have input, output, and some description of the service internals, heterogeneity of ways to define parameters and to describe internal processing (typically done as free text in WSDL) encumbers straightforward integration.

Works that relate to service integration have focused on various aspects of the problem. Semantic Web services were proposed to overcome interface heterogeneity. Using languages such as OWL-S [1], Web services are extended with an unambiguous description by relating properties such as input and output parameters to common concepts, and by defining the execution 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 [10, 3] to manage the complexity of the problem. In [19] and [18], it is argued that the process of service composition may be of 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. In an attempt to support exploratory composition, an 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.

International Workshop on Semantic Data and Service Integration (SDSI'07), September 23, 2007, Vienna, Austria

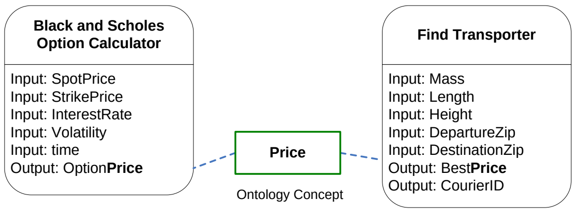


Figure 1: Service Tagging is Misleading: Example

In this work we target an essential element that we believe is still missing in existing approaches. In many cases, services are only partially annotated semantically. Therefore, the matching of services (*e.g.*, for the purpose of composition) may yield results that are semantically incorrect, especially in exploratory scenarios, where the matching is not tightly controlled. As an example, consider two real-world Web services, illustrated in Figure 1. The two services, *Black and Scholes option calculator* and *find transporter* share some common concepts, such as the *price* concept. These two services originate from very different domains. The first is concerned with finance and the second with transportation. It is unlikely that these two services will be combined into a meaningful composition. This example illustrates that methods that are based solely on the concepts mapped to the service's parameters (such as in [12]) may yield inaccurate results. Therefore, in this work we propose the use of *service categorization*, a process that categorizes a service to a set of concepts (or an ontology) that represents a domain, to improve the matching process. Categorizing services to

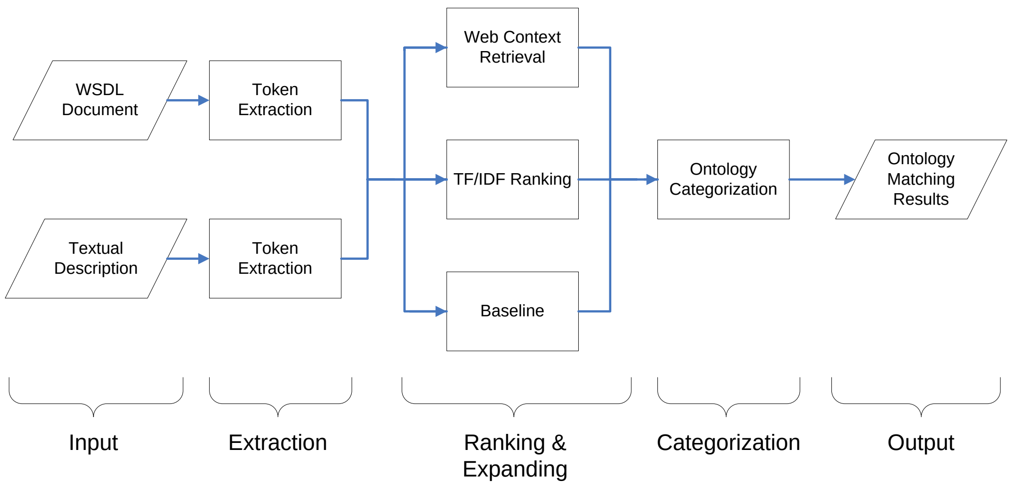


Figure 2: The Web service categorization process

their respective domains can be used to classify the validity of the composition and to rule out compositions of unrelated services.

We provide a preliminary study, comparing the use of two known methods, TF/IDF [16] and context generation [17], in service categorization. We propose a three dimensional classification of methods for associating a Web service with an ontology concept and provide partial experiments to test various combinations from the classification, using a real-world data set. Our analysis indicates that context analysis is more useful than TF/IDF and that WSDL description provides better categorization than textual description.

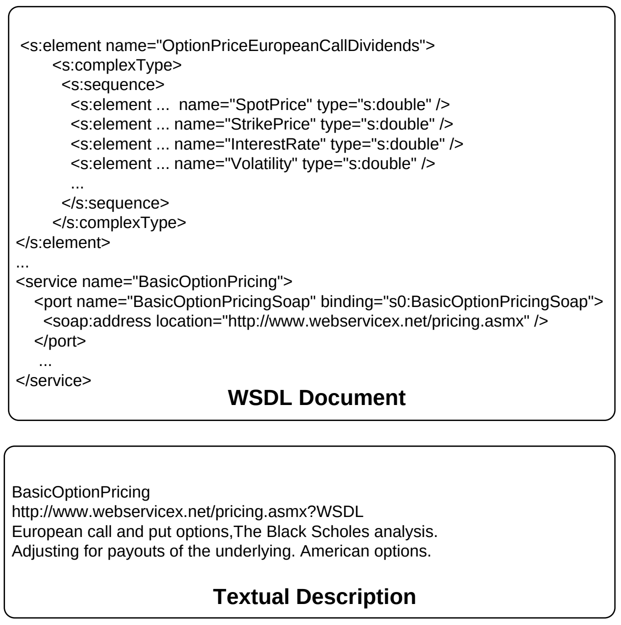
The rest of the paper is organized as follows. Section 2 provides a simple model for service categorization and introduces the three dimensional classification. We next present the setup and results of the experimental evaluation of our work. Section 4 describes related work. Finally, Section 5 concludes the research and provides directions for future work.

2. SERVICE CATEGORIZATION

2.1 Overview

In this section, we specify the process of automatically labeling Web services with semantic concepts for the sake of categorization. We aim at associating an ontology concept to each service. Such a process does not need to be as fine-grained as semantic annotation. For example, a service can be mapped to a whole ontology. Figure 2 depicts the stages of the categorization process, including the different methods we evaluated. We assume that each Web service is described using a textual description (which is part of the meta-data within UDDI registries), and a WSDL document describing the syntactic properties of the service interface. Figure 3 depicts an example of these two descriptions. These descriptions serve as the input to the categorization process.

We examine three methods for the service description analysis: TF/IDF, Web context extraction, and a *baseline* for evaluation purposes. 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 categorization is achieved by matching the bag of tokens to the concept names, attributes and documentation of each of the ontologies.



```
<s:element name="OptionPriceEuropeanCallDividends">
      <s:complexType>
        <s:sequence>
          <s:element ...  name="SpotPrice" type="s:double" />
          <s:element ... name="StrikePrice" type="s:double" />
          <s:element ... name="InterestRate" type="s:double" />
          <s:element ... name="Volatility" type="s:double" />
          ...
        </s:sequence>
      </s:complexType>
</s:element>
...
<service name="BasicOptionPricing">
    <port name="BasicOptionPricingSoap" binding="s0:BasicOptionPricingSoap">
      <soap:address location="http://www.webservicex.net/pricing.asmx" />
    </port>
    ...
</service>
```

WSDL Document

BasicOptionPricing
http://www.webservicex.net/pricing.asmx?WSDL
European call and put options,The Black Scholes analysis.
Adjusting for payouts of the underlying. American options.

Textual Description

Figure 3: The Option Pricing Service: An Example

2.2 Service Analysis

The analysis starts with token extraction, representing each service, S, using two sets of tokens, called *descriptors*. Each token is a textual term, extracted by simple parsing of the underlying documentation of the service. The first descriptor represents the WSDL document, formally put as $D^S_{wsdl} = \{t_1, t_2, \ldots\}$. The second descriptor, $D^S_{desc} = \{t_1, t_2, \ldots\}$, represents the textual description of the service. 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.*, BasicOptionPricingSoap). 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 were expanded as previously mentioned using the capital letter of each word.

In Section 2.6 we shall provide some intuition regarding the differences between the methods of TF/IDF and conext extraction, which serves as a motivation for choosing these two. Nevertheless, this choice is more or less arbitrary. Other methods for text extractions can be used, borrowing from the vast literature of Information Retrieval (IR) [14] and Machine Learning (ML) [11].

Option Price Call Black Scholes
Spot Price
Strike Price
Interest Rate
Volatility
time
Option Price Call Black Scholes Response
Option Price Call Black Scholes Result
Option Price Delta Call Black Scholes
Spot Price
Strike Price
Interest Rate
Volatility

Figure 4: An Example of the Baseline Representation of the Option Pricing Service

2.3 TF/IDF Analysis

TF/IDF (Term Frequency / Inverse Document Frequency) is a common mechanism in IR to generate 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 thrown away 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 $\mathcal{D}_{desc}$ 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 which 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 [15]. 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.

Option
Price
Call
Black
Scholes
Volatility
European
Strike
Spot
Dividend
Interest
Delta
Partials
Dividends
Payout

Figure 5: An Example of the TF/IDF High Scored List of the Option Pricing 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," "Result," and "time" received below-the-threshold TF/IDF weight, due to their high frequency.

2.4 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 which are related to the propositions. The context recognition algorithm was adapted from [17] 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*, $\mathcal{C}$, is assembled.

Calculator
Glossary
Black-Scholes
Financial
Formula
Finance
Trading
Option
Pricing
Model
Analysis
Scholes
Model

Figure 6: An Example of the Web Context

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 [9] 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 had removed the word "price," which appeared in the baseline and the TF/IDF token lists, and introduced the word "pricing," which is more relevant to the financial 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.

2.5 Ontology Matching

The final step of the labeling process is to match 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, with the baseline and the TF/IDF analysis, the two services mentioned in Section 1, the option pricing service and the transportation finding process, were mapped to the same ontology. Using context analysis, they were matched separately to the financial ontology and transportation ontology, respectively. The method of ontology matching is described next.

2.6 Discussion

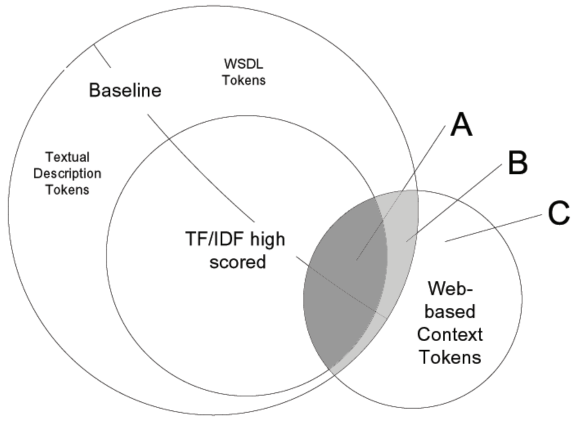


Figure 7: Token sets generated by the analysis methods, and their inter-relations

The service categorization described in the service description provided by the service developer and service provider are inefficient in specifying the categorization. This is due to the perspective of the developer and the terms he uses. Another problem is that the developer is not aware of all the existing ontologies and all of their concepts when providing a service. Furthermore, the provider cannot be forced to supply a detailed description. These textual descriptions usually contain a bare minimum of information which sometimes does not add to the understanding of the service.

Figure 7 depicts the relationships between the sets of tokens produced by the different analysis methods, *i.e.*, TF/IDF and Web context extraction. The larger circle represents all the tokens extracted from the textual description and the WSDL document. We define their union as the *baseline set.* As the TF/IDF provides a mere ranking of the original tokens, it is all contained in the baseline set. The *TF/IDF high scored* set represents all the tokens which received a higher score than a given threshold. The tokens which are the result of the context extraction method are part of the *Web-based context* set. The method identifies existing baseline tokens, and also finds new words, based on a core of the baseline tokens. Therefore, the set overlaps with the baseline set, containing new tokens which were not part of the baseline.

We now provide some insights regarding the various elements of the diagram. The shaded part marked "**A**" is the overlap of both methods. It contains all tokens which belong to the Web context set and to the TF/IDF high scored. Both TF/IDF and the context analysis methods have decided that these certain keywords are relevant for categorization. In our experiments, about 7% of the terms in the context analysis belong to this part. This overlap may serve as evidence of the importance of these keywords. For example, categorizing a service that monitors a workflow process had yielded a very short token list. However, the token "workflow" appeared in the "A" set, as it is unique enough to receive high TF/IDF weight, and relevant enough for Web search to be retrieved as part of several extractions.

The shaded, moon-shaped part marked "**B**" involves those keywords in the baseline that TF/IDF deems irrelevant, while the context analysis method believes otherwise. This part, according to our experiments, constitutes 3% of the keywords returned by the context analysis. Tokens such as "message," "request," and "response" are typical members of this set, since they are frequent words in the WSDL document, they are sometimes retrieved by the context extraction algorithm. Thus, this set can be used to filter out the context.

Part "**C**" marks the great advantage of the context analysis over the TF/IDF method. While the latter has to work within the limits of the baseline dictionary, the context analysis method goes out to the Web, using it as an external judge, returning keywords that are deemed relevant, although they were not originally specified in the baseline description. 90% of the returned keywords belong to this region. Some are indeed evaluated as important while others less so. For example, the descriptor of a service which handles calculation of financial derivatives, was augmented with words such as "trading" and "pricing" which are useful for categorizing the service under the "finance" domain.

We intend to further investigate the inter-relationships between the two methods. For example, instead of analyzing the text in parallel, we can start with TF/IDF, eliminating low score tokens and then processing them with the context analysis method. To get the best of both worlds, we need to combine these methods in a way that they will overcome each other's errors. Therefore, we can combine the tools at our disposal along three dimensions. One dimension determines the relevant description (WSDL vs. textual description, in our case). A second dimension chooses whether to pre-filter or not (baseline vs. TF/IDF filtering). The third dimension chooses the level of extracted Web context that is used (no context, context that overlaps with the TF/IDF set, and pure context). We leave the research into the interrelationships in and between these dimensions open for future research.

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 the number of top ranking results from each cycle of the algorithm. The context method performance execution time is higher than the TF/IDF due to the need to access the Web search engine for every line of input extracted from the WSDL and can reach between to 3-4 minutes for very long WSDL documents. However, since each web service only needs to be classified once in its life time, performance is less crucial than accuracy.

3. EMPIRICAL ANALYSIS

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

3.1 Experimental Setup

3.1.1 Data

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 [8], which includes over 40 different ontologies from various domains.

The experiments compared four different methods for categorization, as described in Section 2:

Description Context The Context Extraction algorithm described in Section 2.4 was applied to the textual description of the Web services. Each descriptor of the Web service context was used as a token.

Name Context The Context Extraction algorithm 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 three methods, with an addition of a **baseline**, which included the original token

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

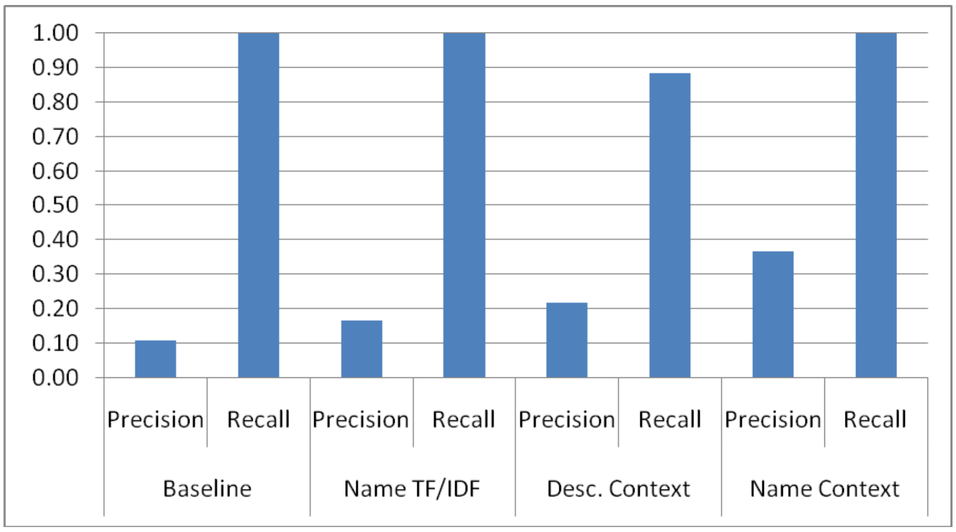


Figure 8: Precision and Recall of All Methods

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 2.5.

3.1.2 Performance Measure

The metrics in our experiments were recall and precision. Recall that services can be categorized 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 categorizes 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.

3.2 Experimental Results

In our first experiment we analyze the usefulness of going beyond the baseline bag of tokens. Recall that in Figure 7 we described the overlap between the baseline and the context tokens. As mentioned before, in our experiments there was an overlap of 10% in the number of tokens. The remaining 90% of the context was an extension to the baseline bag of tokens. 31% of those tokens were not matched in any of the ontologies and therefore could not be used for categorization. Of the matched tokens, 61% led to the successful match with an ontology and 39% provided an incorrect matching. These results are encouraging. It indicates that the use of an external judge assists in better categorization. Nevertheless, the method for generating such an external bag of tokens needs to be improved.

Figure 8 compares the precision and recall of all four methods. 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 Description Context managed to double the precision to 22% at the cost of 11% decrease in recall. 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 categorization. Again, more improvement is needed to increase precision even more.

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

A method for improving precision involves the pre-processing of the ontologies themselves. Using the corpus of ontologies, one can apply TF/IDF and filter out common concepts. For example, generic concepts, such as *price* would be replaced with more precise concepts, such as *option-pricing* and *product-price*. This way, categorization can be improved, relying only on truly identifying tokens.

Figure 9 displays the precision vs. recall of the results, 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

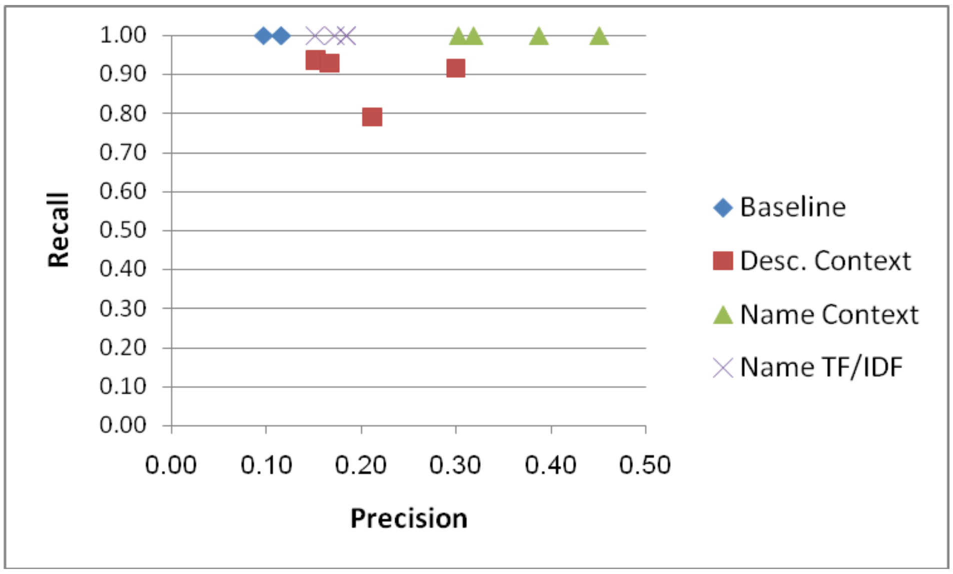


Figure 9: Precision vs. Recall Broken by Methods

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 results of the Description Context method are more dispersed, with a lower recall than all 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 Name Context 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 weakness of the TF/IDF approach is surprising, with respect to its success in other IR fields. There are several explanations for this phenomena. First, the relatively short size of the WSDL document and the service's textual description reduced the effectiveness of TF/IDF. Secondly, as TF/IDF ranks tokens, the original order of the words within the service descriptors was lost. This order was found to be quite important for the ontology categorization process.

Figure 10 displays the precision and recall achieved according to each topic. Note that the symbols in this figure now represent domains and not methods. We clustered each method with circles. The purpose of this figure was to examine whether it is easier for the methods to classify certain topics. Figure 10 shows that in the topics of Business and Communication both the highest and the lowest levels of precision were achieved according to the different methods. The Name Context method is the only method that yielded the highest recall and precision in these two topics compared to the other two topics. The same two topics scored lowest in the Baseline method. *courier services* topic achieved highest results in all methods except for the Name Context method.

We believe that it is easier to classify the *courier services* topic in all methods except for the Name Context method because many of the tokens that appear in the labels can be identified in the ontology, since this is a smaller domain. 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.

4. RELATED WORK

Patil et al. [13] 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. A similar method, which also aggregate results from several matchers, is presented by Duo et al. [5]. 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.

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

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

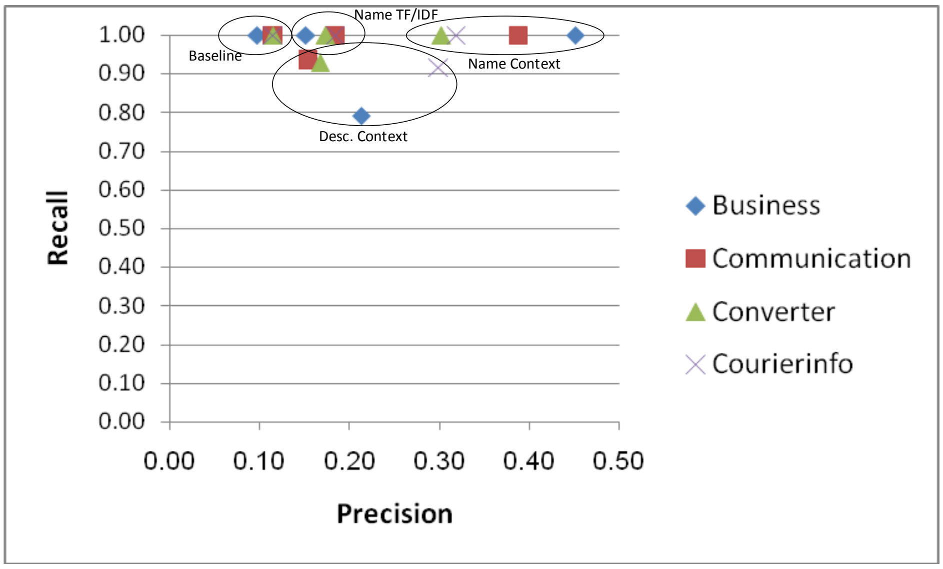


Figure 10: Precision vs. Recall According To Each Topic

their mutual distance in the WSDL file. Our results suggest that one of the main reasons for the inferiority of TF/IDF analysis was explained by losing the original order of the words within the WSDL file.

5. CONCLUSION

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 users with usable methods to investigate and access large scale service repositories. Rather than asking users to manually annotate their services with formal concepts, our method harnesses the information contained in the World Wide Web for establishing 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 categorizing 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 services. Also, we have encountered some problems with the use of general-purpose ontologies. Clearly, categorization will work better with smaller, focused ontologies, and we intend to seek many such ontologies for a more rigorous experimentation. We shall also look at methods for identifying coherent sub-ontologies based on categorization results.

We seek to fully embed the proposed method as a preprocessing step in $\mathcal{OPOSSUM}$ and test the outcome in improving its retrieval capabilities. Other methods for service categorization will be tested as well.

6. 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] M. Gu, A. Aamodt, and X. Tong. Component retrieval using conversational case-based reasoning. In *Proceedings of the International Conference on Intelligent Information Systems (ICIIP 2004)*, pages 21–23, Beijing, China, October 2004.

[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] 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.

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

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

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

[12] 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.

[13] 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.

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

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

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

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

[18] 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.

[19] 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.