Context-Based Matching and Ranking of Web Services for Composition

Aviv Segev Eran Toch

Abstract—In this work we propose a two-step, context-based semantic approach to the problem of matching and ranking web services for possible service composition. We present an analysis of different methods for classifying Web services for possible composition and supply a context-based semantic matching method for ranking these possibilities. Semantic understanding of Web services may provide added value by identifying new possibilities for compositions of services. The semantic matching ranking approach is unique since it provides the Web service designer with an explicit numeric estimation of the extent to which a possible composition "makes sense." First, we analyze two common methods for text processing, TF/IDF and context analysis, and two types of service description, free text and WSDL. Second, we present a method for evaluating the proximity of services for possible compositions. Each Web service WSDL context descriptor is evaluated according to its proximity to other services' free text context descriptors. The methods were tested on a large repository of real-world Web services. The experimental results indicate that context analysis is more useful than TF/IDF. Furthermore, the method evaluating the proximity of the WSDL description to the textual description of other services provides high recall and precision results.

Index Terms—Intelligent Web services and semantic Web, internet reasoning services, Web text analysis

1 Introduction

In recent years, the use of services to compose new applications from existing modules has gained momentum. Web services are autonomous units of code, independently developed and evolved. The Web Service Description Language - WSDL [1] is used as the de-facto standard for service providers to describe the interface of the Web services, i.e., their operations and input and output parameters. Therefore, Web services lack homogeneous structure beyond that of their interface. Heterogeneity stems from different ways to name parameters, define parameters, and describe internal processing. This heterogeneity encumbers straightforward integration between Web services.

Web service registries such as Universal Description, Discovery and Integration (UDDI) were created to encourage interoperability and adoption of web services. However, UDDI registries have some major flaws [2]. UDDI registries either are made publicly available and contain many obsolete entries or require registration. In either case, a registry only stores a limited description of the available services.

Semantic Web services were proposed to overcome interface heterogeneity. Using languages such as Ontology Web Language for Services (OWL-S) [3] and WSDL Semantics (WSDL-S) [4], 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* [5], which serve as the key mechanism to globally define and reference concepts. Formal languages enable service composition, in which a developer uses automatic or semi-automatic tools to create a integrated business process from a set of independent Web services.

Service composition in a heterogeneous environment immediately raises issues of evaluating the accuracy of the mapping. As an example, consider three real-world Web services, illustrated in Figure 1. The three services - distance between zip codes (A), store IT contracts (B), and translation into any language (C) - share some common concepts, such as the *code* concept. However, these three services originate from very different domains. Service A is concerned with distance calculation and uses the *zip codes* as input, service B defines *CurrencyCode* as part of the IT contract information to be stored, and service C uses a *ClientCode* as an access key for users. It is unlikely that any of the services will be combined into a meaningful composition. This example illustrates that methods based solely on the concepts mapped to the services parameters (as in [6]) may yield inaccurate results.

We aim at analyzing different methods for automatically identifying possible semantic composition. We explore two sources for service analysis: WSDL description files and free textual descriptors, which are commonly used in service repositories. We investigate two methods for Web service classification for each type of descriptor, Term Frequency / Inverse Document Frequency (TF/IDF) [7] and context-based analysis [8], and a baseline method. We define contexts as a model of a domain

- *A. Segev is with the Department of Management Information Systems, National Chengchi University, Taipei 116, Taiwan. E-mail: asegev@nccu.edu.tw*
- *E. Toch is with the School of Computer Science, Carnegie-Mellon University, 5000 Forbes Avenue, Pittsburgh 15218, United States. E-mail: eran@cs.cmu.edu*

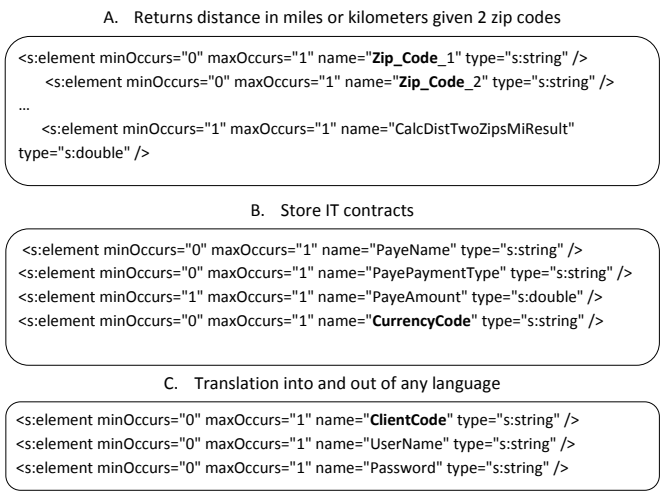


Fig. 1. Service Tagging is Misleading: an Example

for a given term, which is automatically extracted from a fragment of text. In this work, contexts are created by finding related terms from the Web. Unlike ontologies, which are considered shared models of a domain, we define contexts as local views of a domain [9]. Therefore, contexts may be different for two fragments of information, even though their domain might be the same. The definition of context used throughout this paper extends the definition of context in ubiquitous computing, which employs context as any information that can be used to characterize the situation of an entity [10]. In many fields, context is used to describe the environment in which a service operates. In our definition, it is used to describe the related set of linguistic terms of a given text.

In this work we propose a context-based approach to the problem of matching and ranking semantic Web services for composition. First, we propose the use of service classification, a process that matches a service to a set of concepts, representing its affinity with a given domain. For example, consider the services in Figure 1. The context of service A would be a set of geographical terms (such as address, city, and distance). Therefore, it would be classified to a set of concepts taken from a geographical ontology. Service B would be classified to a business transaction ontology and service C to a computer systems ontology. Second, we use the classification and context information to improve the process of service composition, ruling out compositions of unrelated services. Given a suggested composition between a number of services, we analyze the context overlap between the services. The overlap is used to rank the probability of the composition.

Our method is particularly relevant to exploratory composition rather than automatic composition. An exploratory composition [11] is an iterative process in which an automatic composer suggests possible options for the composition and a human user decides regarding the final composition. Thus, the process is suitable for situations in which semantic annotations may be incomplete and incorrect. Our method can be used to rank the possible candidate services for composition, which is resilient to partial annotations.

To evaluate our method, we conducted a set of experiments in which we compared our method to other methods, including pure TF/IDF and simple string matching. We used a real-world data set, containing WSDL description and textual description of public Web services. Our results show that the Web based context extraction method analyzing both the WSDL description and the textual description yields better results than the TF/IDF method and string matching. In addition, the results prove the advantage of integrating the analysis of both the WSDL context descriptor and the service textual descriptor. Furthermore, to test whether the model can be used to provide useful recommendations of Web service compositions, each Web service context extracted from the WSDL is compared with all other Web service contexts extracted from the textual description. The experiments examine the classification of two points of view, the analyzed "internal" view, the given WSDL description, and the "external" view, the text description of the different services from which we are looking for possible compositions. The numerical estimation evaluates what the distance is between the "internal" view and the "external" view. The results provide high recall and precision for the Web context extraction based method compared to other methods. In addition, we show that the proximity analysis of the context of each Web service can serve as a useful tool for suggesting Web service composition.

The rest of the article is structured as follows. Section 2 presents an analysis of the service classification methods. Section 3 describes the service composition proximity analysis method. Section 4 discusses the differences between the different methods. Section 5 presents experiments with matching Web services based on the different service analysis methods and experiments with the composition proximity analysis method. Section 6 describes the related work. Section 7 highlights our main contribution and presents future work.

2 Service Categorization for Composition

2.1 Overview

In this section, we specify the process of automatically labeling Web services with semantic concepts for the sake of categorization. Previous work on Semantic Annotations for WSDL and XML Schema (SAWSDL) [12] defined how to add semantic annotations to various parts of a WSDL document such as input and output message structures, interfaces, and operations. 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

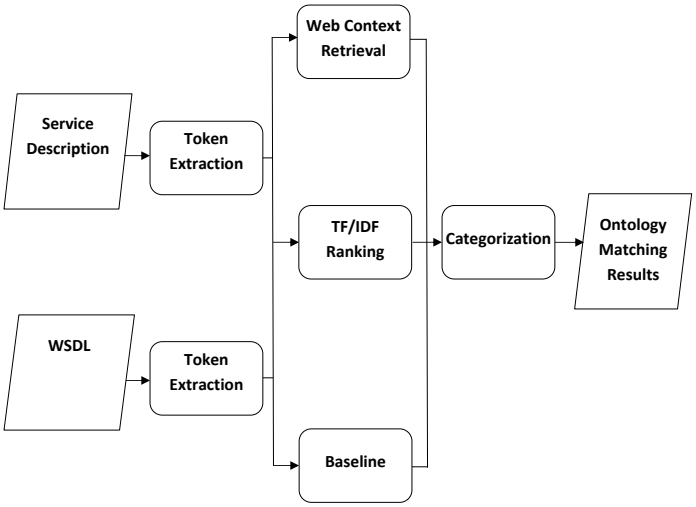


Fig. 2. The Web Service Categorization Process

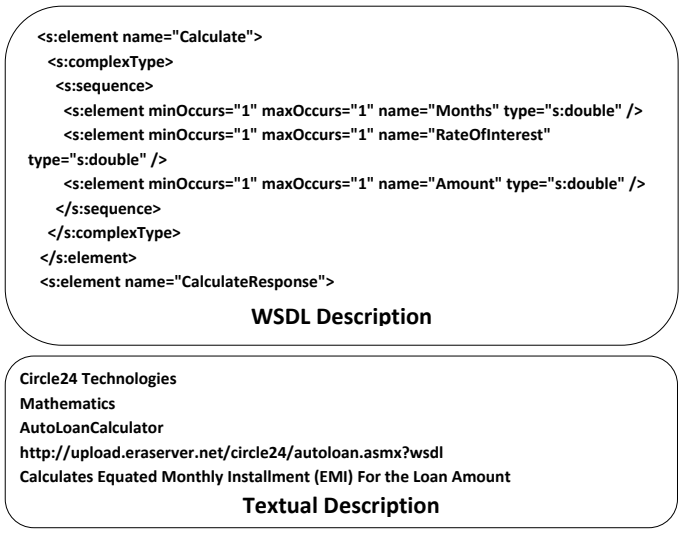


Fig. 3. An Example: Calculates Equated Monthly Installment for Loan Amount

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 classification analysis: TF/IDF, Web context extraction, and a *baseline* for evaluation purposes. The baseline method is a simple reflection (identity function) of the original bag of tokens, extracted from the service descriptions to a bag of tokens representing sets of words. The process of service analysis, which leads to the construction of the baseline, is described in Section 2.2. The basic data structure used by all the methods is a ranked bag of tokens, which is processed and updated in the different stages. The results of the service analysis process are used by the TF/IDF and Web context extraction methods, as described in Section 2.3 and Section 2.4. After the different analysis methods were applied, the final categorization is achieved by matching the bag of tokens to the concept names of each of the ontologies.

2.2 Service Analysis

The analysis starts with token extraction, representing each service, S, using two sets of tokens, called *descriptors*. Relying on pre-defined WSDL documentation tags was initially considered for evaluation but found to be less valuable since often Web service providers do not include the tags in their service [13]. Each token is a textual term, extracted by the 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 and is supplied by the service provider in free text. 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.*, CalcDistTwoZipsMiResult). Therefore, the tokens are divided into separate tokens. 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, including tokens that might be less relevant. The sequence of words was expanded as previously mentioned using the capital letter of each word. The tokens are filtered using a list of *stopwords*, removing words with no substantive semantics.

In this section we describe the two methods used for the classification of Web services, TF/IDF and context extraction. Section 4 provides some intuition regarding the differences between the methods of TF/IDF and context extraction, which serves as a motivation for choosing these two. Nevertheless, this choice is more or less arbitrary. Other methods of text extraction can be used, borrowing from the vast literature of Information Retrieval (IR) [14] and Machine Learning (ML) [15].

2.3 TF/IDF Analysis

TF/IDF is a common mechanism in IR for generating 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. 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

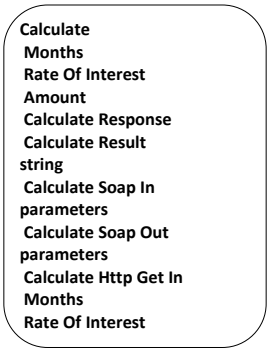


Fig. 4. Initial Processing Example of the Equated Monthly Installment for Loan Amount

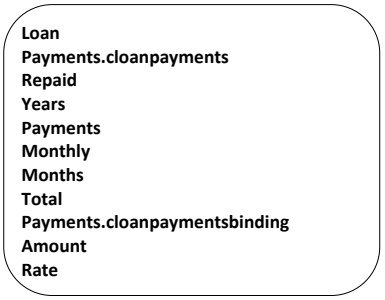


Fig. 5. An Example of the TF/IDF High Scored List of the Equated Monthly Installment for Loan Amount

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 chose 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 [16]. 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.

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 "Body," "String," and "Post" received below-the-threshold TF/IDF weight, due to their high inverse document frequency.

2.4 Context Extraction

We define a descriptor c_i from domain $\mathcal{DOM}$ as an index term used to identify a record of information [17], which in our case is a Web service. It can consist of a word, phrase, or alphanumerical term. A weight $w_i \in \Re$ identifies the importance of descriptor c_i in relation to the Web service. For example, we can have a descriptor $c_1 = Mortgage$ and $w_1 = 36$. A *descriptor set* $\{\langle c_i, w_i\rangle\}_i$ is defined by a set of pairs, descriptors and weights. Each descriptor can define a different point of view of the concept. The descriptor set defines all the different perspectives and their relevant weights, which identify the importance of each perspective.

By collecting all the different viewpoints delineated by the different descriptors we obtain the *context*. A *context* $\mathcal{C} = \left\{\{\langle c_{ij}, w_{ij}\rangle\}_i\right\}_j$ is a set of finite sets of descriptors. For example, a context $\mathcal{C}$ may be a set of words (hence $\mathcal{DOM}$ is a set of all possible character combinations) defining a Web service and the weights can represent the relevance of a descriptor to the Web service. In classic IR, $\langle c_{ij}, w_{ij}\rangle$ may represent the fact that the word c_{ij} is repeated w_{ij} times in the Web service descriptor document.

The context recognition algorithm was adapted from [8]. 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 Web service, 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. McCarthy [18] defines a relation $ist(\mathcal{C}, P)$, asserting that a proposition P is true in a context $\mathcal{C}$. In our case the adopted algorithm uses the corpus of WSDL descriptors, $\mathcal{D}^{wsdl}$, and textual description, $\mathcal{D}^{desc}$, as propositions P_i, and the contexts describing the WSDL as descriptors c_{ij} with their associated weight w_{ij}. 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 algorithm input is defined as a set of textual propositions representing a Web service. Each textual proposition is sent to a Web search engine. The set of descriptors is extracted by clustering the Web pages search results. The number of textual propositions from which the same descriptor is extracted identifies the number of references to the descriptor in the text. Similarly, the number of Web pages that identify the same descriptor represents the number of references in Internet documents. A high ranking in only one metric does not necessarily indicate the importance of the context: for example, high ranking in only Internet references may mean that it is an important topic but might not be relevant to the document. To combine both metrics, the two values are weighted to contribute equally to final weight value.

The context recognition algorithm consists of the following major stages: selecting contexts for each text, ranking the contexts, and declaring the current contexts. The result of the token extraction is a list of keywords obtained from the text. The selection of the current context is based on searching the Internet for relevant documents according to these keywords and on clustering the results into possible contexts. The output of the ranking stage is the current context or a set of highest ranking contexts. The set of preliminary contexts that has the top number of references, both in number of Internet pages and in number of appearances in all the texts, is declared to be the current context and the weight is defined by integrating the value of references and appearances.

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 person when determining the set of contexts that describes the current scenario (the Web service in our case). For example, Figure 6 provides the outcome of the Web context extraction. The figure displays the context which includes only the highest ranking descriptors that pass the cutoff to be included in the context.

Fig. 6. An Example of the Web Context

The use of the Internet as a context database instead of a precalculated frequencies base [19] has several advantages. The use of the Internet does not require the constant updating and maintenance of a database, while the precalculated frequencies base requires the user to work in a limited predefined knowledge domain. Also, the Internet can serve as an unlimited knowledge domain that is continuously being updated. Although the Internet contains some unreliable Web sites, their influence can be overcome when a large quantity of Web sites is analyzed. Last but not least, the multilingual nature of the Internet makes it a perfect infrastructure for integrating Web services described in different languages or cultural perspectives. The context representing a service can be composed of descriptors represented in different languages.

An advantage of the Web context extraction approach over simple token matching methods (TF/IDF and baseline) is the ability to add new possible contexts, textual descriptors, of the Web service which do not appear in the original text. This ability derives from the use of the Internet as a knowledge base for collecting the possible descriptors, while other methods limit the descriptors to keywords appearing in the WSDL. For example, the Web context extraction described in Figure 6 includes the descriptor "Mortgage", which did not appear in the original WSDL description displayed in Figure 4, and extends the concept of "Loan" from a financial domain to other domains, including the business domain (to which the service actually belongs).

A context can consist of multiple descriptor sets and can be viewed as a meta-representation of the Web service. The added-value of having such a meta-representation is that each descriptor set can belong to several ontology concepts simultaneously. For example, a descriptor set $\{\langle\ Calculator, 2\rangle\}$ can be shared by many ontology concepts that have interest in computational analysis (such as economic forecasting, sales representatives, simple mathematical analysis, etc.) although it is not in their main role definition (and hence the low weight assigned to it). Such overlap of contexts in ontology concepts influences the task of Web service composition. The appropriate interpretation of a Web service context that belongs to several ontology concepts is that the service is relevant to all such concepts. Therefore, it can be considered for composition with each of the Web services belonging to the same concepts. This leads to the possibility of service composition based on different perspectives of the service use.

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.

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 three services mentioned in Section 1, the distance in miles or kilometers service A, the store IT contracts service B, and the translation service C, were mapped to the same ontology. With context analysis, they were matched separately to different ontologies.

Flexibility of mapping context to ontology with respect to language has been proposed in [20]. Multilinguality requirements necessitate the adaptation of the ontology to different languages separately. Avoiding such multiple efforts is desirable, both for the initial specification of the ontology and for the ontology evolution. Here, the context can serve as the translation mechanism, according to which ontological concepts are interpreted in the local language. Each ontology concept can be represented by multiple contexts in different languages. To illustrate this point, consider the English concept *loan*, representing a concept in the field of finance. While in English the concept can be represented with one word, in French the concept would require two contexts: *emprunt* (used when borrowing) and *prêt* (used when lending). The use of contexts to also represent the ontological concept (such as loan) compensates for any under-specification that may result from the universality of the ontology.

3 Service Ranking for Composition

The analysis of the ranking of services for possible composition is based on the advantage that a Web service can be separated into two descriptions: the WSDL and a textual description of the Web service in free text. The Web service WSDL descriptor and the Web service textual descriptor have different purposes. One describes "how" the service should be used and the other describes "what" the service does. However, they both describe the same service from different perspectives. If we are looking for possibilities of service composition, the motivation of the comparison lies in the investigation of how this service can be expanded in comparison to what other services can do.

The section analyzes the possible compositions of each pair of Web services. The goal is to supply a ranking, a numeric estimation of the complementary relation between each two Web services. Due to the large number of possible service combinations, it is impractical to check all the options manually. The evaluation can provide the Web service designer with a ranking that will identify which services should be considered first. The overlap between each pair of service textual description context will be analyzed versus other Web services WSDL context. This will form the analysis for each service, which will allow a two-way context proximity comparison. The proposed method yields a numeric estimation of the extent to which a composition should be considered. Furthermore, the method can suggest unique compositions due to the idea that the Web service analysis is based on semantic, rather than syntactic, meaning.

Figure 7 displays the process of service ranking based on context analysis to suggest possibilities of composition. The process integrates the advantages of both top ranking methods in the previously described categorization analysis. The process includes the following core components: initial analysis, Web context retrieval, and analysis of possible composition. Similar to the previous section, when analyzing the possibilities for Web services composition, 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. These two descriptions, as described in Figure 3, serve as the input to the analysis process. The initial processing step is similar to both the textual service description and the WSDL and includes token extraction and stop-list words. The second step includes applying the context extraction method for both the service textual description and the WSDL, resulting in a context which is composed of a descriptor set. Both the WSDL processing and the context extraction are fully automated. The two descriptors sets for each Web service are separate inputs for the composition analysis step, which is the following key component.

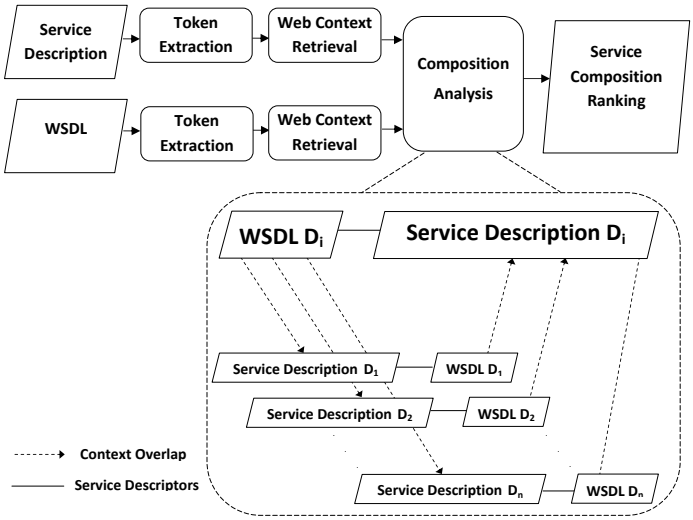


Fig. 7. Service Composition Context Analysis Process

To consider a composition of Web services, we analyze the relation of the context of each service to other services. We analyze the context of each service in a bi-directional method. Each Web service context is evaluated according to its proximity to other services and the proximity of each of the other services to the current service. Figure 7 on the bottom part displays an enlargement of the bi-directional service description context composition analysis. The composition analysis

compares each Web service WSDL descriptor context with all other Web service textual descriptor contexts. The output of the process is a ranked list of pairs of Web services which could be considered for composition.

The results of the mapping of the Web services to onotologies in Section 2 serve as a prior stage to the current process. Previously the Web services were mapped to ontologies represented by concepts and their relations. The current stage can select a single concept as input for composition or a set of related concepts. Alternatively, the process can run on the entire ontology to suggest possible compositions which were not preconsidered. However, using the option of the entire ontology increases the chance of composition suggestions of unrelated services.

The following sections describe the initial analysis, followed by a short review of the context extraction process described earlier. The last section details the main component, the context overlap analysis for possibilities of service composition.

3.1 Initial Analysis

The initial analysis is similar to the process described in Section 2.2. 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 $D^{wsdl} = \{t_1, t_2, \ldots\}$. The second descriptor, $D^{desc} = \{t_1, t_2, \ldots\}$, represents the textual description of the service. All WSDL elements classified as *name* were extracted as tokens. WSDL tokens require special handling: the tokens are divided into separate tokens using the capitalized first letter of each word. The tokens are filtered using a list of *stop-words*, removing words with no substantive semantics. The result of this stage is the baseline token list depicted in Figure 4.

3.2 Context Extraction

The next stage includes extracting the context of the WSDL descriptor and the service textual description using the method described in Section 2.4. The extraction process uses the Web as a knowledge base to extract multiple contexts for the tokens. Extraction is used to filter out biased tokens, provide a more precise ranking, and extend the service descriptors. The context recognition algorithm consists of the following major stages: initial processing, context retrieval, context ranking, and context selection. The selection of the context is based on searching the Internet for relevant documents according to these descriptors and on clustering the results into possible contexts. The last step includes ranking the descriptors and selecting the top ranking set which defines the context of the service. The process is performed on both the WSDL description of the Web service and the textual description of the Web service. The output of this step is similar to the results displayed in Figure 6.

3.3 Analysis of Possible Service Composition Using Context Overlap

To analyze a set of Web services and identify which services are more likely to be composed, we analyze the overlap between the services based on their context. We compare each Web service WSDL descriptor context with another Web service textual descriptor context, as described in Figure 7.

Let $WS = \{D_1, D_2, ..., D_n\}$ define a set of Web services descriptions which are analyzed for possible composition. We denote Context Overlap (CO) as:

$$CO(D_i^{wsdl}, D_j^{desc}) = |\{c_k \epsilon D_i^{wsdl} \bigcap c_k \epsilon D_j^{desc}\}|$$

which defines the number of context descriptors, c_k, of Web service D_i WSDL descriptor that overlaps with context descriptors of Web service D_j textual descriptor. Similarly, for $CO(D_j^{wsdl}, D_i^{desc})$ we calculate the overlap in reversed order. Notice that two different sets of descriptors are analyzed in both comparisons. $COMP(D_i, D_j)$ computes the composition likelihood, the proximity, between two given Web services based on their weighted Context Overlap:

$$COMP(D_i, D_j) = \sqrt{CO(D_i^{wsdl}, D_j^{desc})^2 + CO(D_j^{wsdl}, D_i^{desc})^2}$$

All the Web services pairs in the WS set are evaluated for composition likelihood. All the Web services pairs, which are not identical, are evaluated according to their context descriptor set D_i. $COMP(WS)$ computes the two services from a given set with the highest likelihood of composition to be:

$$COMP(WS) = Max_j\left(Max_i(COMP(D_i, D_j))\right) \forall i, j \mid D_i, D_j \in WS, i \neq j$$

The suggested method can identify different services which yield the same functionality. In this case the developer might decide to unite the two services to supply multiple interfaces such as input or output variable definitions to the same functionality.

The method can be used iteratively to identify the top n services which could be considered for composition. Alternatively, a threshold could be used for the $COMP$ value to define the number of compositions which should be considered. The proposed method does not perform the composition but rather suggests which of the Web services should be considered and prioritizes a list of possible compositions. Although the method provides the developer with a ranking of which services should be considered for composition based on semantic similarity, the developer is still required to resolve the structural integration related to the Web service structural or syntactical issues.

The output of the context overlap process to identify possible compositions is a ranking of which services should be considered for composition. Figure 8 displays textual descriptions for some of the high ranking

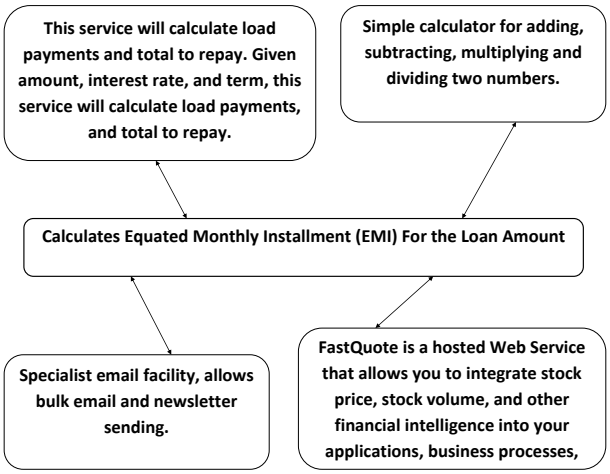


Fig. 8. Possible Services Identified for Compositions with Equated Monthly Installment for Loan Amount

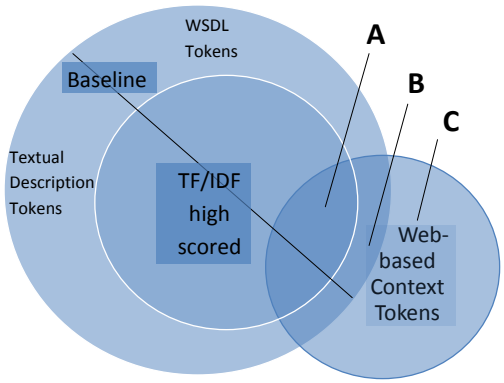


Fig. 9. Token Sets Generated by the Analysis Methods and Their Inter-Relations

possible compositions identified by the algorithm for composition based on semantic meaning. Figure 3 included WSDL descriptions of three of the same services and emphasized the difficulty of identifying composition based on syntactic meaning.

4 Discussion

The service categorization, which is part of the service description provided by the service provider, is insufficient in fully specifying the categorization, due to the provider's perspective and the terms he uses. Another problem is that the provider 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 contain a bare minimum of information which sometimes does not add to the understanding of the service.

Figure 9 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 entirely 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 set. Both TF/IDF and the context analysis methods 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 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 part marked "**B**" contains 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 as a filter to remove from the context words which were given a low ranking by TF/IDF and a high ranking by 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 knowledge source, 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 are 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.

Further investigation of the inter-relationships between the two methods can be performed. 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 shortcomings. 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 prefilter 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 inter-relationships in and between these dimensions open for future study.

The context overlap method can be used to identify possibilities of Web services composition. The method not only analyzes whether the two services should be considered for composition but also supplies a numeric estimation of the extent to which a composition can be made in comparison to the other services.

The effort required to analyze all possible combinations for a large quantity of Web services is time consuming. The numeric estimation of the extent of the composition allows a Web service engineer to prioritize the analysis of each possible composition. The ability of the algorithm to evaluate the composition according to the context based semantic matching approach that uses the Web as a knowledge source extends the role of Web services in new directions. The algorithm thus integrates two points of view, the "internal" view, the given WSDL description, and the "external" view, the text description of the different services from which we are looking for possible compositions. The numerical estimation evaluates what the distance is between the "internal" view and the "external" view.

Previous work presented a model and a set of algorithms to semi-automatically support relationship evolution in an ontology using contexts [21]. The motivation for this work stems from the difficulty in supporting ontology evolution. An ontology "bootstrapping" approach can be developed based on analyzing the Web services using the different methods for description analysis. Each method can represent a different perspective of viewing the Web service. Two methods can be used to invoke new possible concepts. A third method can be used to resolve inconsistencies with the current ontology. An ontology evolution can be performed if all analysis methods agree on the identification of a new concept or a relation change between the ontology concepts. However, this issue remains 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 that of TF/IDF due to the need to access the Web search engine for every line of input extracted from the WSDL. However, since each web service only needs to be classified once in its lifetime, performance is less crucial than accuracy.

5 Empirical Analysis

In this section we describe our experiments and provide empirical analysis and comparison of the different methods of the classification and possible composition of services.

5.1 Experimental Setup

5.1.1 Data

The data for the experiments were taken from an existing benchmark repository provided by researchers from University College Dublin.[1] Our experiments use a set of 392 Web services, originally divided into 20 different topics such as: courier services, currency conversion, communication, business, etc. For each Web service the repository provided a WSDL document and a short textual description.

The ontologies used for comparison were selected to represent three topics described in Figure 10. The sets of ontologies were selected as possible topics into which the Web services could be classified. The additional constraint on selecting the sets of ontologies was that there exist multiple ontologies rich enough to represent each of the topics. The first set of ontologies was selected as general ontologies and is supposed to encompass a representation of everything in the world. This set included the SUMO and the Mid-Level-Ontology. The second set of ontologies was on Web Finance and included the Finance the Web and Economy ontologies. Both these sets of ontologies were taken from the OWLS-TC [22] ontology repository. The last set of ontologies included a collection of existing ontologies collected via Web search to represent the domain of Entertainment. These ontologies include the Music, Game, Home Environment, and Device ontologies.

The next step included a manual classification of all the Web services into the three possible topics. Each

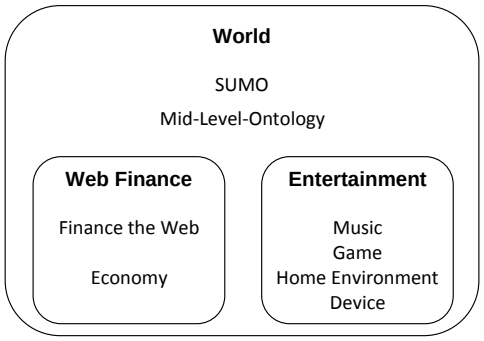


Fig. 10. Ontology Sets

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

Web service could belong to more than one topic. According to the classification, there were 392 web services belonging to the World, 167 to the Finance, and only 27 belonging to the Entertainment ontology set.

5.1.2 Classification Methods

The experiments examined four different methods for service classification, as described in Section 2. The service classification methods included:

1) **WSDL Context** The Context Extraction algorithm described in Section 2.4 was applied to the *name* labels of each Web service. Each descriptor of the Web service context was used as a token.
2) **WSDL TF/IDF** Each word in the document was checked for term frequency and inverse document frequency (TF/IDF) as described in Section 2.3. The set of highest ranking weighted value words was used.
3) **Description Context** The Context Extraction algorithm was applied to the textual description of the Web services. Each descriptor of the Web service context was used as a token.

The first set of experiments compared the three methods and 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 ontology string matching method described in Section 2.5.

The second set of experiments analyzed the service composition ranking. The analysis measured the overlap between the WSDL Context and Description Context of each of the services. The results were compared to the overlap of the token list extracted from services using the baseline and the TF/IDF methods. The comparison was based on the method for evaluating overlap and composition described in Section 3.3.

5.2 Web Services Classification Results

In our first experiment we analyze the usefulness of going beyond the baseline bag of tokens. Figure 11 compares the recall and precision of the methods of classification. The results are displayed according to each of the ontology sets. The X-label describes each of the methods and the Y-label describes the level of recall and precision.

The first World ontology set is supposed to encompass all possible web services. The baseline method achieved only 91.58% recall. The lowest classification for the World set was 90.05% from the TF/IDF method. The results indicate that the WSDL document does not have enough information defined as *name* tokens to be self descriptive. The result of 94.90% was achieved by the WSDL Context method, which can be attributed to the external tokens added to the baseline method. Furthermore, the WSDL description based on free text description of the service achieved 98.98% recall. The WSDL descriptor results emphasize even further the need for external description in addition to the basic web service descriptive language. The precision in this case is 100% in all methods since all of the services belong to the World topic.

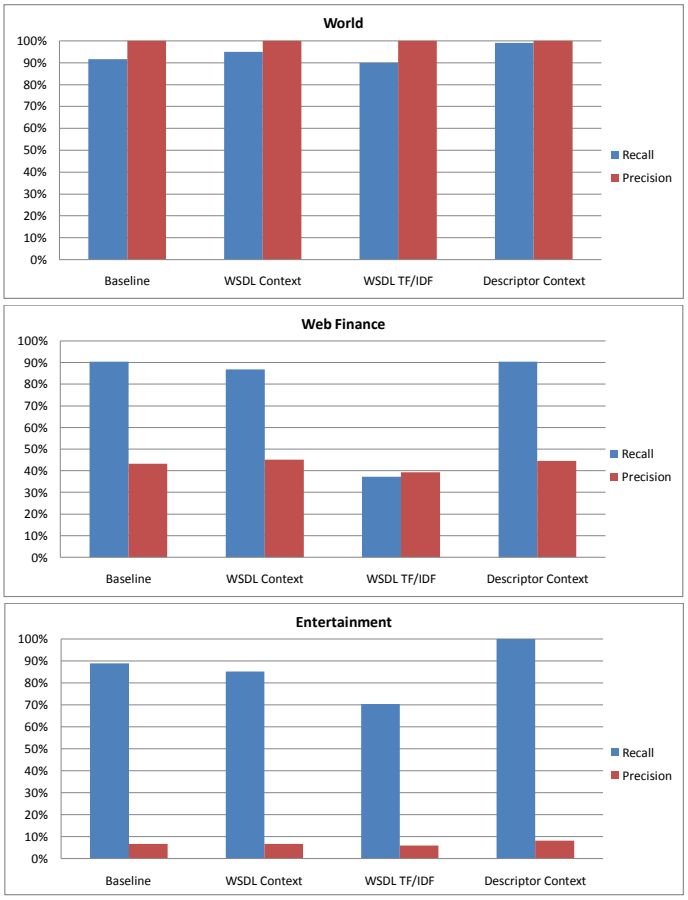


Fig. 11. Precision and Recall of All Methods

The Web Finance classification precision shows a slight advantage again to the WSDL Context method of 45.17%, followed closely by the WSDL Descriptor Context precision of 44.54%, over the baseline 43.14% and the TF/IDF 39.24% methods. However, in this case there is a drop in the recall of the WSDL Context to 86.83% versus the higher recall result of the baseline 90.42%. The lower recall result of the TF/IDF 37.13% can be explained due to the limited number of general recurring tokens. In this case, the retrieval achieved the same high results as the baseline 90.42%. Examination of the recall results between the baseline and the TF/IDF points to the high recall based on terms with high frequency which are not topic specific. The high retrieval rate and higher precision in both context based methods in this case again indicate the importance of the external Web description.

The Entertainment classification achieved very low precision rates. This could be attributed to the Music ontology, which was relatively big and diversified, and as a result many of the web services were identified mistakenly as relevant. The baseline achieved higher

recall results 88.89% than the WSDL context 85.19% and TF/IDF 70.37%. Again we can see the importance of the external description of the WSDL, which outperformed the other methods, achieving 100.00% recall and 8.21% precision. The precision results of the baseline and WSDL Context were almost identical, 6.65% and 6.63% respectively, followed by the TF/IDF precision of 5.85%.

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.

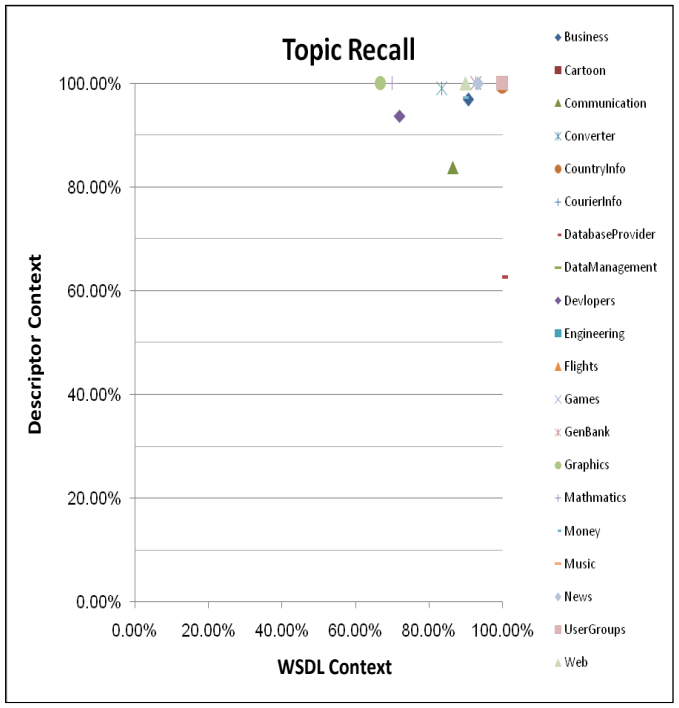
Figure 12 displays the recall and precision, partitioned into topics, of the best performing methods WSDL Context and Descriptor Context. The graph analyzes the advantages and disadvantages of each descriptor according to topics. The results are presented on a X-Y method comparison graph, where classification according to the WSDL Context method is given on the X-axis and classification according to the Descriptor Context is given on the Y-axis. All 20 original classification topics are displayed in the legend. The top graph presents the average recall of the three ontology sets for the different topics and the bottom represents the average precision of the ontology sets for the topics.

We see that low WSDL Context and high Descriptor Context recall results were achieved by Graphics, Mathematics, Developers, and Converter topics. This could be attributed to the lack of matching ontology sets that could fit the proper topics. Database-Provider obtained the opposite results with high WSDL Context and low Descriptor Context recall. The result is due to the low quality description accompanying these web services. The Communication topic received relatively lower results in both methods, although not the lowest results. The results suggest that the Communication topic is more diverse and thus harder to classify.

Analysis of the precision graph indicates that low WSDL Context and Descriptor Context results were achieved by Converter, News, CountryInfo, Mathematics, and Developers. The Converter, Mathematics, and Developers low precision and low recall results can be attributed to the service descriptor as being too broad and lacking specific matching ontology. However, for News and CountryInfo, which have high recall and low precision results, it can be inferred that the Economy and Music sets of ontologies do not define these topics conclusively enough. Notice also that the precision of the topics is mostly aligned with the central diagonal which shows similar results for both methods in most topics.

The results indicate that two main issues impact the WSDL and textual description results:

1) The quality of the textual description associated with the web service. This indicates that the web service WSDL by itself is **not** enough to classify a Web service. The results emphasize the importance of adding sufficient descriptors to the WSDL when supplying the service for public use.
2) The ability to select proper ontology sets which classify correctly the web services is of secondary importance. To increase the recall and precision results a small set of ontologies should be selected. The ontology size itself also plays a role since a larger ontology has a greater chance of lowering the precision.



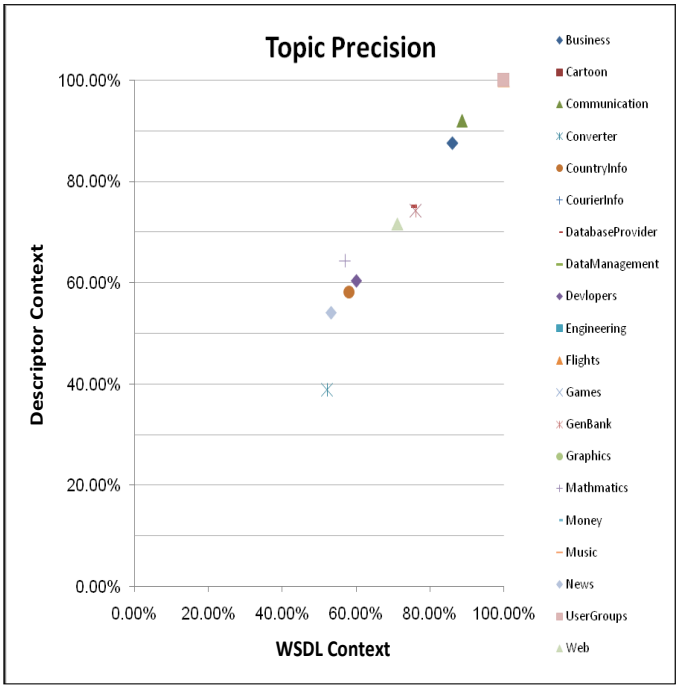


Fig. 12. Precision vs. Recall According to Each Topic

5.3 Ranking Web Services for Composition Results

The Web services composition analysis used the set of 392 Web services previously described in Section 5.1.1. Similar to the previous experiments, for each Web service the repository provided a WSDL document and a short textual description. The purpose was to analyze the quality of the possible composition according to the context overlap method described in Section 3.3. The analysis of the set of Web services and the identification of which services would be more likely to be composed are performed by analyzing the overlap between the services based on their context. We compare each Web service WSDL descriptor context with another Web service textual descriptor context. This integrates the advantages of both of the top ranking methods, WSDL Context and Description Context.

The performance of the integrated context method was compared to that of the Baseline and TF/IDF methods. For the Baseline and the TF/IDF methods the overlap method was used to analyze the precision. For the TF/IDF the list of terms which result after processing the algorithm was analyzed for overlap.

The top ranking pairs for composition of each of the methods were compared. To analyze the recall and the precision for each of the methods, the top ranking pairs were hand-labeled as relevant or irrelevant. An ideal result for a recall versus precision graph would be a horizontal curve with high precision value; a bad result has a horizontal curve with a low precision value. The recall-precision curve is considered by the IR community the most informative graph showing the effectiveness of the results.

Figure 13 displays the results of the Context Overlap, TF/IDF Overlap, and Baseline Overlap precision for the top ranking Web services identified as composition pairs. The horizontal X-axis displays the total number of Web services pairs considered. The vertical axis displays the precision value for each composition analysis method.

The results indicate that the Context Overlap achieved a precision level of 100% when the number of top ranked Web services pairs considered was 14, 24, and 37. When the set of the top ranked Web services is 54, the precision drops to 96.29%. The precision went down slightly more to 94.32% when 88 top ranking Web services were analyzed. Finally, for 151 Web service compositions the precision reached 88.08%.

When comparing the Context Overlap to the Baseline Overlap results, a difference of almost 30% is maintained in the precision throughout the graph. The initial top ranking 14 results for the Baseline achieved 71.43%, slowly dropping to 58.94% for 151 service pairs. The TF/IDF graph displays a fluctuating behavior, starting from 57.14% at 14 pairs, reaching a peak of 70.83% at 24 pairs, and dropping twice and ending with a precision value of 59.33% at 151 pairs. The TF/IDF fluctuation displays worse results than does the Baseline. This suggests that the frequency of the words in the original text does not improve the identification of possible compositions of Web services. However, the high precision results achieved by the Context Overlap display the importance of using the Web as an external knowledge source.

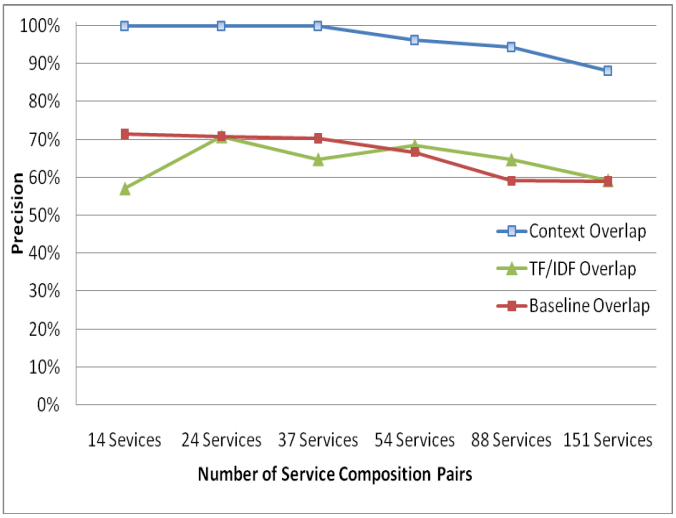


Fig. 13. Precision of Service Matching According to Contextual Ranking

In the example displayed in Figure 8 the Web service which *Calculates Equated Monthly Installment (EMI) For the Loan Amount* located in the center is identified for composition with the two top services of *Calculate load payments and total to repay* and *Simple calculator* for the Context Overlap of 37 Web services. Similarly, the composition of the EMI service is identified for composition with the *Specialist email facility* and *FastQuote* for Context Overlap of the top 151 Web services composition.

Figure 14 displays the recall versus the precision of all three methods. In the graph the X-axis represents the recall and the Y-axis the precision.

The graph shows that the Context Overlap methods supply the highest recall precision values throughout the graph. As the recall increases over 50%, the precision drops from 100% in the beginning and ranges between 53% and 78%. The TF/IDF precision drops from the beginning much faster and when the recall increases, the precision drops to 14%. The Baseline method displays low recall-precision during most of the values.

The graph displays the advantage of the integrated context method in comparing the WSDL description to the textual description of all other services. The integrated context method dominates, in both recall and precision, both the TF/IDF and the Baseline methods.

6 Related Work

The field of Web service composition is very active. However, most approaches require clear and formal semantic annotations to formal ontologies [23], [6], [4], [22]. As most services which are currently active in the World Wide Web do not contain any semantic annotations, finding methods that enable composition with-

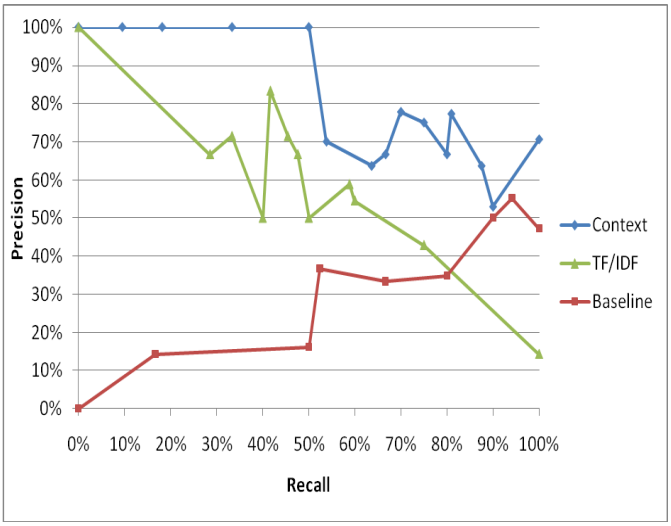


Fig. 14. Recall vs. Precision Overlap Comparison

out semantic annotation is a necessity. Initial work has been done in discovering services directly by querying syntactic Web services through their WSDL documentation ([24] and [25]). Our work provides an analysis of different ways for extracting information from syntactic Web services and using this information in the context of composition, rather than Web service discovery.

The field of automatic annotation of syntactic Web services contains several works relevant to our research. Patil et al. [26] presented a combined approach towards automatic semantic annotation of Web 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. [27] present a similar method, which also aggregates results from several matchers.

Oldham et al. [28] showed that using a simple machine learning technique, namely Naïve Bayesian Classifier, improves the precision of service annotation. Machine learning is also used in a tool called Assam [13], which uses existing annotation of semantic Web services to improve new annotations. While machine learning effectively improves the efficiency of the semantic annotation, the corpus size used for learning is small, as WSDL documents contain very little text. Our approach is complementary to machine learning methods, as it suggests and provides further information, in the form of textual descriptions and Web context. This information can be used by learning methods to improve annotations.

Another relevant field is search engines for syntactic Web services. Works by Platzer and Dustdar [2] and Woogle by Dong et al. [29] present search engines for WSDL documents. The search engines use a multitude of information retrieval techniques, including vector space representation, TF/IDF, and text clustering. The main drawback of applying these techniques to WSDL is the relatively short content of a WSDL document, which limits the precision and recall of the search engine. Our work explores methods for overcoming this problem by using Web context and service descriptions.

More recently, several works suggested to use information about the Web service composition to provide a better annotation process. Bowers and Ludäscher [30] proposed to explore the relation between input and output parameters of the same operation to infer the semantics from the parameters. If the semantics of the input parameter is known, and the logic of the operation is known, then the semantics of the output parameter can be inferred automatically. Belhajjame et al. [31] suggest to use information about the composition (the term workflow is used in their work) in which the service is used. The composition structure reveals operational constraints between parameters of different operations and can be used to support or disqualify annotations.

The aforementioned works by Bowers and Ludäscher and Belhajjame et al. show the potential of using external information for improving annotations. Our work shares a similar vision, arguing for the utilization of external information. However, our intention is to produce domain specific semantic annotation rather than operational semantics. Therefore, we use the Web and public ontologies as information resources, rather than the workflow or procedural description of the Web services.

Context-based semantic matching for Web services composition has become a focus of interest. An initial prior work describes a context mediator that facilitates semantic interoperability between heterogeneous information systems [32]. A recent work presents a context-based mediation approach [33] which was used to solve semantic heterogeneities between composed Web services. However, these approaches require that the context definition be manually tailored, as opposed to automatically generated, for each set of services.

An initial analysis of different methods for mapping Web services to ontologies based on text processing using the Web as an external source was presented in [34]. The analysis was performed on a small set of services which is currently extended. The initial analysis yielded different results due to the limited data sources used. In addition, the present paper proposes a method for ranking each possible composition, allowing developers to prioritize the work involved in the service composition.

7 Conclusions

The ability to compose Web services based on WSDL and free text descriptions can potentially simplify the implementation of business processes. 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 Web for establishing rich context for user queries.

Our experiments prove the inherent problems of analyzing WSDL documents. Their short length and limited vocabulary pose serious challenges for labeling

and categorizing services. The weak performance of the TF/IDF measure, which works successfully on more verbose texts, shows that relying on the service text alone will not yield adequate results. The proposed method has several limitations that will be addressed in future research. These limitations include explicit handling of multilingual service description, analyzing implications of selecting specific Web sites for context extraction, and analyzing implications of different Web context extraction methods. Additional directions of research include an analysis of other options of context overlap for possible compositions, such as context overlap based only on textual description overlap, and the matching of textual description to WSDL, the reverse operation to the one in the current research.

REFERENCES

[1] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "WSDL web services description language," http://www.w3.org/TR/2001/NOTE-wsdl-20010315, W3C, W3C Candidate Recommendation, 2001.

[2] C. Platzer and S. Dustdar, "A vector space search engine for Web services," in *Proceedings of the Third European Conference on Web Services (ECOWS05)*, 2005.

[3] A. Ankolekar, D. Martin, Z. Zeng, J. Hobbs, K. Sycara, B. Burstein, M. Paolucci, O. Lassila, S. Mcilraith, S. Narayanan, and P. Payne, "DAML-S: Semantic markup for web services," in *Proceedings of the International Semantic Web Workshop (SWWS)*, July 2001.

[4] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. T. Schmidt, A. Sheth, and K. Verma, "WSDL-S web service semantics," http://www.w3.org/Submission/WSDL-S/, W3C, W3C Candidate Recommendation, 2005.

[5] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein, "OWL web ontology language reference," http://www.w3.org/TR/owl-ref/, W3C, W3C Candidate Recommendation, 2004.

[6] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *Proceedings of the International Semantic Web Conference*, 2002.

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

[8] A. Segev, M. Leshno, and M. Zviran, "Context recognition using internet as a knowledge base," *Journal of Intelligent Information Systems*, vol. 29, no. 3, pp. 305–327, 2007.

[9] A. Segev and A. Gal, "Putting things in context:a topological approach to mapping contexts to ontologies," *Journal of Data Semantics (JoDS)*, vol. IX, 2007.

[10] A. K. Dey, *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, Georgia Institute of Technology, 2000.

[11] J. Rao, D. Dimitrov, P. Hofmann, and N. Sadeh, "A mixed initiative approach to semantic web service discovery and composition: Sap's guided procedures framework," in *ICWS '06: Proceedings of the IEEE International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 401–410.

[12] J. Farrell and H. Lausen, "Semantic annotations for wsdl and xml schema SAWSDL," http://www.w3.org/TR/2007/REC-sawsdl-20070828/, W3C, W3C Candidate Recommendation, 2007.

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

[14] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. London: Butterworths, 1979.

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

[16] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for idf," *Journal of Documentation*, vol. 60, no. 5, pp. 503–520, 2004.

[17] C. Mooers, *Encyclopedia of Library and Information Science*. Marcel Dekker, 1972, vol. 7, ch. Descriptors, pp. 31–45.

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

[19] R. Chau and C. Yeh, "A multilingual text mining approach to web cross-lingual text retrieval," *Knowledge-Based Systems*, vol. 17, pp. 219–227, 2001.

[20] A. Segev and A. Gal, "Multilingual ontology-based knowledge management," *Decision Support Systems*, vol. 45, pp. 567–584, 2008.

[21] A. Segev and A. Gal, "Ontology verification using contexts," in *Proceedings of ECAI-Workshop on Contexts and Ontologies: Theory, Practice and Applications*, 2006.

[22] 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.

[23] S. C. Oh, "Effective web-service composition in diverse and large-scale service networks," Ph.D. dissertation, University Park, PA, USA, 2006, adviser-Soundar R. Kumara.

[24] G. A. Vouros, F. Dimitrokallis, and K. Kotis, "Look ma, no hands: Supporting the semantic discovery of services without ontologies." in *SMRR*, ser. CEUR Workshop Proceedings, R. L. Hernandez, T. D. Noia, and I. Toma, Eds., vol. 416. CEUR-WS.org, 2008.

[25] E. Toch, A. Gal, and D. Dori, "Automatically grounding semantically-enriched conceptual models to concrete web services," in *ER*, ser. Lecture Notes in Computer Science, L. Delcambre, C. Kop, H. Mayr, J. Mylopoulos, and O. Pastor, Eds., vol. 3716. Springer, 2005, pp. 304–319.

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

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

[28] N. Oldham, C. Thomas, A. P. Sheth, and K. Verma, "Meteor-s web service annotation framework with machine learning classification," in *SWSWPC*, 2004, pp. 137–146.

[29] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Simlarity search for web services." pp. 372–383, 2004.

[30] S. Bowers and B. LudŁscher, "Towards automatic generation of semantic types in scientific workflows," in *In Proceedings of the International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS), WISE 2005 Workshop Proceedings, LNCS*, 2005, pp. 207–216.

[31] K. Belhajjame, S. M. Embury, N. W. Paton, R. Stevens, and C. A. Goble, "Automatic annotation of web services based on workflow definitions," *ACM Trans. Web*, vol. 2, no. 2, pp. 1–34, 2008.

[32] E. Sciore, M. Siegel, and A. Rosenthal, "Using semantic values to facilitate interoperability among heterogeneous information systems," *ACM Transactions on Database Systems*, vol. 19, no. 2, pp. 254–290, 1994.

[33] M. Mrissa, C. Ghedira, D. Benslimane, Z. Maamar, F. Rosenberg, and S. Dustdar, "A context-based mediation approach to compose semantic web services," *ACM Transactions on Internet Technology*, vol. 8, no. 1, p. 4, 2007.

[34] 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.