International Journal of Web Information Systems

Automatic categorization of Web service elements

Efthimia Mavridou, Konstantinos M. Giannoutakis, Dionysios Kehagias, Dimitrios Tzovaras, George Hassapis,

Automatic categorization of Web service elements

Efthimia Mavridou
Electrical and Computer Engineering Department, Aristotle University of Thessaloniki, Thessaloniki, Greece

Konstantinos M. Giannoutakis, Dionysios Kehagias and Dimitrios Tzovaras
Information Technologies Institute, Centre for Research and Technology Hellas, Thermi, Thessaloniki, Greece, and

George Hassapis
Electrical and Computer Engineering Department, Aristotle University of Thessaloniki, Thessaloniki, Greece

Received 28 August 2017
Revised 28 August 2017
Accepted 3 January 2018

Abstract

Purpose – Semantic categorization of Web services comprises a fundamental requirement for enabling more efficient and accurate search and discovery of services in the semantic Web era. However, to efficiently deal with the growing presence of Web services, more automated mechanisms are required. This paper aims to introduce an automatic Web service categorization mechanism, by exploiting various techniques that aim to increase the overall prediction accuracy.

Design/methodology/approach – The paper proposes the use of Error Correcting Output Codes on top of a Logistic Model Trees-based classifier, in conjunction with a data pre-processing technique that reduces the original feature-space dimension without affecting data integrity. The proposed technique is generalized so as to adhere to all Web services with a description file. A semantic matchmaking scheme is also proposed for enabling the semantic annotation of the input and output parameters of each operation.

Findings – The proposed Web service categorization framework was tested with the OWLS-TC v4.0, as well as a synthetic data set with a systematic evaluation procedure that enables comparison with well-known approaches. After conducting exhaustive evaluation experiments, categorization efficiency in terms of accuracy, precision, recall and F-measure was measured. The presented Web service categorization framework outperformed the other benchmark techniques, which comprise different variations of it and also third-party implementations.

Originality/value – The proposed three-level categorization approach is a significant contribution to the Web service community, as it allows the automatic semantic categorization of all functional elements of Web services that are equipped with a service description file.

Keywords Semantic web, Machine learning, Web services

Paper type Research paper

1. Introduction

Given the growing popularity and increased presence of web services (WS), efficient mechanisms and tools are required to facilitate their matchmaking, publishing, discovery and semantic annotation. Service categorization is considered to be an essential step to facilitate automatic matching and semantic annotation of WS. The results of service categorization may be used for service matchmaking purposes and semantic annotation by

International Journal of Web Information Systems
Vol. 14 No. 2, 2018
pp. 233-258

1744-0084
DOI 10.1108/IJWIS-08-2017-0059

relevant applications, such as semantic annotation frameworks that transform non-semantic WS to semantic ones. Our work contributes to fulfilling the need for more accurate automatic service matching and semantic annotation support tools by introducing an automatic categorization technique for non-semantic WS, i.e. those services, for which a machine understandable meaningful context is not available.

Our WS categorization technique extends the work of Giannoutakis *et al.* (2015) and Mavridou *et al.* (2012) and presents an advanced three-level categorization scheme which categorizes semantically a non-semantic WS description and all its elements. Specifically, it predicts the application domain to which a WS belongs and categorizes WS operations and input and output parameters into existing descriptive classes. The application domains with the corresponding descriptive classes ("ideal" operations) are semantically described and annotated within an ontology.

Our work focuses on how to efficiently deal with the large number of classes of the WS classification task by using Error Correcting Codes (ECOC) (Hamming, 1950; Mavridou *et al.*, 2012). The latter is used as a means for decomposing a multi-class classification problem into many binary classification ones and finally combines all results to decide upon the correct class. Based on empirical evaluation, our work ascertains that the use of the ECOC increases the classification accuracy of the WS mechanism, especially for the operation categorization task, where the number of classes is too large, e.g. compared to the domain categorization task.

Besides applying some text preprocessing techniques to extract terms from WS descriptions, such as stemming, we use the simplicity of the Bayes formula as a vectorizer for transforming the feature space to a lower dimension representation (Mavridou *et al.*, 2012). Reduction of the feature space dimension is required primarily because an increased number of features requires an increased computational effort by classification-based techniques that are primarily used by the majority of WS categorization techniques.

Moreover, by reducing the feature space, noisy features are often eliminated; hence, the quality of the classification accuracy increases (Yang and Pedersen, 1997). The aforementioned techniques are used in conjunction with Logistic Model Trees classifier (LMT) (Landwehr *et al.*, 2005), which combines logistic regression with tree induction, for efficiently dealing with noisy data. LMT has the advantage to rely on simple regression models if only a few and/or noisy data sets are available and adds a more complex tree structure if there is enough data to warrant such structure. Finally, our work deals with the categorization of i/o WS parameters by the use of bipartite graphs for semantic similarity matching along with Wordnet dictionary and the Hungarian algorithm (Kuhn, 1955). The advancement that is introduced in this respect is that our approach combines in an innovative fashion, lexicographic, data type and structure-related characteristics to calculate the semantic similarities of WS i/o parameters. The three levels of categorization (domain, operation and i/o) are integrated into a single framework for automatic WS categorization that comprises the main contribution of this paper. Based on empirical evaluation, the proposed mechanism results in increased accuracy of the overall WS categorization process, compared to a set of known approaches.

2. Related work

Our work is focused on semantic categorization, which can be applied to non-semantic WS. However, it also addresses some aspects of service annotation and discovery. These are described in the next subsections.

2.1 Semantic annotation of services

Various ontology frameworks have been proposed for adding semantics to WS (OWL-S (Martin *et al.*, 2007), WSMO (Roman *et al.*, 2005), SAWSDL (Verma and Sheth, 2007), WSMO-Lite (Vitvar *et al.*, 2008) and WSDL-S (Akkiraju *et al.*, 2005). However, most of the WS found on the Web do not contain semantic information. To annotate them, significant manual effort is required. The manual process is time-consuming, and its complexity increases exponentially as the number of WS and ontology concepts increase.

The most common WS representation format that fails to include semantics is the XML-based, Web Service Description Language (WSDL) (Chinnici *et al.*, 2007), which describes WS whose communication relies on the SOAP protocol. In this case, the WSDL document provides a mechanism for services to advertise their capabilities to clients. As opposed to this, RESTful WS lack a formal means of advertising themselves to their clients. Nevertheless, a similar to WSDL representation has been proposed (W3C submission) (Hadley, 2006) for RESTful services, known as WADL (Web Application Description Language).

In this case, WADL (Web Application Description Language) can be used as an equivalent representation to WSDL for RESTful services. Relevant research efforts are mostly focused on the definition of formalisms for creating semantic annotations for RESTful services (Fang *et al.*, 2012; Kopecky *et al.*, 2008; Maleshkova *et al.*, 2009; Sheth *et al.*, 2007). An interesting approach that relates to our work is presented in (Khorasgani *et al.*, 2011), which creates the WADL files from the HTML documentations of RESTful APIs and deploys linguistic heuristics for matching those files (Kopecky *et al.*, 2008).

2.2 Service categorization

Particular information retrieval (IR) techniques that are used in the context of WS categorization include text retrieval, vector modeling and clustering.

Machine learning (ML) techniques are suggested as an adequate alternative because schema matching is computationally extensive (Oldham *et al.*, 2005). In our work, we make limited use of schema matching techniques only for discovering similarities between complex data types. Another similar tool, Assam (Heß *et al.*, 2004) uses an ensemble ML approach for WS classification. To the best of our knowledge, METEOR-S Web Service Annotation Framework (MWSAF) (Oldham *et al.*, 2005; Patil *et al.*, 2004), and Assam represents the most well-known integrated tools for WS categorization that are publicly available.

Feature selection and feature transformation methods can be used for reducing the dimension (Yang and Pedersen, 1997) of the vector space of features. In our approach, we use the Naïve-Bayes classifier for transforming the feature space to a lower-dimension representation. The classifier is robust enough to ignore serious deficiencies in its underlying naive probability model and its robustness is encapsulated into the relative magnitudes of the probabilities. We use the simplicity of the Bayes formula as a vectorizer (Isa *et al.*, 2008). The evaluation results show that this approach increased the predictive accuracy of our classifier.

Some notable ML-based techniques have been recently proposed for WS categorization. Among these, Lexicon-based alignment (Canturk and Senkul, 2011) finds the associations between WS and ontology concepts by comparing the different semantics of each term found in a WS document. This approach requires one-by-one comparisons between all different terms, thus resulting in a large number of comparisons. One approach, which can be considered similar to ours, creates features from a large-scale taxonomy (i.e. ontology) used as data for training an SVM classifier

and selects the most dominant features based on Wordnet similarity, for reducing the feature space dimension (Wang *et al.*, 2010). This method, as well as the ones presented in Bruno *et al.* (2005), He *et al.* (2010), Katakis *et al.* (2009) and Sharma *et al.* (2014), serve adequately the needs for service categorization into application domains, but fail to support classification of WS operations.

Further relevant approaches use clustering techniques for the definition of the different classes to which a WS provisionally belongs. An approach of this kind is presented in (Liang *et al.*, 2009). Its advantage is that it does not require training data. However, it shows a relatively low performance when tested with *real* WS (i.e. precision-recall rate almost 60 per cent tested with 352 WSDL documents). Another approach (Hasan *et al.*, 2014) discusses clustering of WS into domains based on QoS, whereas Chen *et al.* (2013) groups together services with similar functionalities using user-contributed tagging data. Furthermore, Alfazi *et al.* (2015) applies clustering techniques for the categorization of cloud services using semantics and cosine similarity metrics.

Additional clustering-based approaches include Chen *et al.* (2013), Hasan *et al.* (2014) and Li and Yang (2012). In all those cases, clustering-based service categorization implies the calculation of similarities between vectors that represent WS and all instances included in each cluster. This process is typically computationally expensive, especially as the feature space dimension grows up. In addition, the WS classes generated by clustering techniques are not specifically accurate, as they may fail to provide clearly distinguished and separable clusters in some cases.

A recent research effort (Mustafa and Swamy, 2015) proposed a multi-layer neural network classification scheme that makes use of metaheuristic search methods. The experimental results are positive but are achievable to only large taxonomies with multiple hierarchical levels.

2.3 Service discovery

The outcome of service categorization techniques can be used in the context of service discovery. Part of these works use techniques such as schema matching (Hao and Zhang, 2007; Sellami and Boucelma, 2011). Hao *et al.* (2010) use IR and schema matching techniques for computing the relevance and importance of services, while Wang *et al.* (2011) use fragmental semantic data to enhance services discovery, which is also useful in the context of service categorization. For instance, Sellami and Boucelma (2011) aim to correctly classify a WS in common cases, such as when two WS have similar functionality, but totally different structure. Similarly, service categorization techniques aim to match a service to a given application domain. In our work, we also leverage categorization of operations to a given semantically annotated context focusing on the functionality of WS encapsulated in the names of operations, rather than their structure.

3. The Web service categorization framework

3.1 Pre-processing mechanism for extracting features from Web service descriptions

Our automatic WS semantic categorization framework receives a collection of WS description documents D as input. Data pre-processing procedure for the training data set describes the data pre-processing procedure that is applied to the initial data set, whose purpose is to prepare the data by removing words included in a stop-word list and by applying a stemming algorithm.

3.1.1 Data pre-processing procedure for the training data set

```
Data-Pre-Processing(D) : V
01. D: WS collection; V: feature vector
02. V ← null;
03. {D_tr, D_ts} ← split_sets(D);
04. For each w_i ∈ D_tr
05.   S ← parse(w_i)
06.   {S1, ..., Si} ← split_words(S)
07.   S' ← stopwords(S)
08.   S'' ← stemming(S')
09. End for
10. For each S_m ∈ S''
11.   If V ∩ {s_m} = ∅
12.     V ← V ∪ {s_m}
13.   End if
14. End for
15. Return V
```

The output of the pre-processing function is a set V of distinct words.

Initially, D is split into training and test sets by the application of function *split_sets()* on data set D. Splitting is performed in one of the two following ways: we select a specific number of random instances within the original data set to form two concrete, non-overlapping tests and training subsets, and we apply the leave-one-out cross-validation technique (Sammut and Webb, 2011) to create the test data set. Based on the latter technique, a single instance from the original data set is used for testing, whereas the remaining instances are used as training data. This process is repeated for each instance that occurs in the original data set.

Set S is generated as a "bag of words" after all available WSDL/WADL documents w_i are parsed (Data pre-processing procedure for the training data set, line 05), and their features regarding operations, i/o parameters are extracted. From the various elements that comprise a WSDL/WADL document only the operations, along with their corresponding input and output parameters are taken into account because these elements include all necessary information that is relevant to the operational characteristics of the corresponding WS. Also, additional information provided by the optional <*documentation*> and <*doc*> tag that encompasses human-readable documentation in-side any part of the WSDL/WADL documents respectively is also taken into account. The terms that are extracted from WSDL and WADL files are outlined in Table I. The main difference between the two formats is that methods in WADL are organized in resources. A resource element can contain a set of methods or other resources. It is identified by a URI that follows a common pattern. However, as it is shown in Table I, all common elements, i.e. service name, documentation, operation, input and output, which are supported by both formats include the most significant information that is necessary for describing the functionality of a WS, especially the operation/method tag.

Function *split_words()* is then applied (Data pre-processing procedure for the training data set, line 06) to the extracted bag of words for splitting them into distinct tokens. Splitting is performed in different ways for operation and i/o elements. In particular, when a word corresponds to an operation name, the most appropriate naming convention among the ones used by WS developers is taken into account,

Table I.
WSDL and WADL elements extracted during the pre-processing stage

WSDL tag	WADL tag	Description
<service>	<resources>	Provides a unique name or URI
–	<resource>	Describes a set of resources, each identified by a URI that follows a common pattern
<documentation>	<doc>	Human readable documentation
<operation>	<method>	SOAP/HTTP method that can be applied to each resource
<message>	–	Envelope element of unique name
<part>	–	A logical abstract content of unique name
<portType>	–	A named set of abstract operations
<input>	<request>	Input to operation as a collection of parameters
<output>	<response>	Output that results from performing an operation
–	<param>	Describes a parameterized component
–	<option>	One of a set of possible parameter values
–	<representation>	A child of a request, response or application element

whereas a different one is assumed when it comes to i/o parameters. Generally, a valid operation name includes strings written in camel case or strings using the underscore character ("_") to join separate words. The generated operation name usually describes the functionality of the operation in a developer-readable fashion. Taking this into account, we apply string manipulation functions, to extract those joint words from operation and i/o parameter names.

In the next stage (Data pre-processing procedure for the training data set, Lines 07-08), all the extracted tokens are filtered so that only unique elements remain. First, all words are filtered by means of a stop-word list. The stop-word list contains articles, prepositions, WS-related words and generally words that appear frequently in WS documents, and thus are not discriminated (the words *the*, *a*, etc.). Furthermore, words that correspond to HTML tags or Web links (which are often found within the documentation tags) are also removed. Generally, removing stop words is considered a necessary step for filtering non-relevant terms (Patil *et al.*, 2004).

The next step involves removal of inflectional endings by applying the Porter stemmer algorithm (Porter, 1980) that results in reducing words to their stems. The unique features that remain after the pre-processing actions comprise the feature set (vocabulary) $V = [t_0, t_1, \ldots, t_{|V|}]$ that is used for representing each WS as a vector.

3.2 Reducing the feature set by using the Bayes' theorem

The feature set V that was generated as the output of the previous pre-processing procedure represents the training data set. Specifically, the vector space model (VSM) is used to represent each WS as a vector $ws_i = [f_{i0}, f_{i1}, \ldots, f_{i|V|}]$, with f_{ik} being the frequency of occurrence of term t_k for each feature k. The same representation is also used for the operation categorization task.

Term weighting schemes such as tf-idf (Salton and Buckley, 1988) are also valid and could be used instead of term frequency. However, although such techniques vectorize the data easily, the number of dimensions is equal to the number of features (Isa *et al.*, 2008). The number of the extracted features may typically range from a few tens to one hundred, thus causing an overfitting effect that has a negative impact on the performance of the overall procedure. In general, the most typical task required for reducing the number of extracted

features is to select only a subset of all features according to some feature selection criterion (e.g. based on the chi-squared statistic). Feature selection techniques use a scoring function to assign a score to each feature, so that only those features with the highest scores are maintained among all features. Although feature selection techniques generally reduce the complexity of the most common classification algorithms, thus increasing their accuracy, the effect of term removal, as a consequence of feature selection, is to increase the risk of removing potentially useful information (Sebastiani, 2002).

To avoid this, we propose an alternative feature dimension reduction technique, which preserves all extracted terms that result after the application of the pre-processing procedure. The goal of this technique is to transform the training data to a lower dimension format, to reduce the number of feature dimension in question, by using the Bayes' theorem. This approach seems to improve the WS classification performance, in terms of required training time and classification accuracy, when used with a support vector machine (SVM) classifier for text categorization (Isa *et al.*, 2008). Based on this, our technique adopts the use of the Bayes' theorem in conjunction with an LMT classifier to improve the accuracy of WS categorization.

The classification of a new example relies on the application of the Bayes' theorem for selecting the class that is most likely to have generated that example (Mccallum and Nigam, 1998). There are several variations of Naive Bayes models. The most common ones are the multivariate Bernoulli model, the Poisson Naive Bayes model and the multinomial model (Eyheramendy *et al.*, 2003). The most apparent difference between these models is that the Poisson model and the multinomial model use feature occurrence frequencies, whereas the Bernoulli model uses binary occurrences (Sebastiani, 2002). In our work, we apply the multinomial Naive Bayes model because it has shown the best performance on conducting text classification (Mccallum and Nigam, 1998).

In the context of WS classification, the domain c to which a WS ws_i belongs is determined as the one for which the probability $P(c_j|ws_i)$ that ws_i belongs to the domain c_j, has its maximum value. $P(c_j|ws_i)$ is calculated by the application of the Bayes' theorem, as shown in equation (1). The term $P(ws_i|c_j)$ is equal to the probability that for a given domain c_j all features of ws_i occur in that domain. $P(c_j)$ is the probability that ws_i belongs to the domain c_j, whereas $P(ws_i)$ is the probability of occurrence of ws_i:

$$P(c_j|ws_i) = \frac{P(ws_i|c_j)P(c_j)}{P(ws_i)} \tag{1}$$

$P(c_j)$ is calculated by equation (2) as the number $N_{t\in c_j}$ of features in the domain c_j, divided by the total number of features in the training data set $N_{t\in D_{tr}}$. It is not necessary to calculate the value of $P(ws_i)$ because it is always fixed. $P(ws_i|c_j)$ is eventually calculated by equation (3) as the product of all probabilities of each feature t_k appearing in c_j:

$$P(c_j) = \frac{N_{t\in c_j}}{N_{t\in D_{tr}}} \tag{2}$$

$$P(ws_i|c_j) = \prod_{k=0}^{|V|} P(t_k|c_j) \tag{3}$$

Each probability $P(t_k \mid c_j)$ is calculated by equation (4) as the number $n_{t_k \in c_j}$ of occurrences of the feature t_k in the domain c_j divided by the total number of occurrences of all features in c_j:

$$P(t_k|c_j) = \frac{n_{t_k \in c_j}}{\sum_{m=0}^{|V|} n_{t_m \in c_j}} \tag{4}$$

An alternative way of computing $P(t_k \mid c_j)$ is by deploying the Laplacian smoothing operation (Friedman *et al.*, 1997), as it is shown in equation (5). The basic idea is to add a constant term both to the numerator and denominator of equation (4) to smooth the estimation of $P(t_k \mid c_j)$ in case a feature that does not occur in the training set occurs in the test set only:

$$P(t_k|c_j) = \frac{1 + n_{t_k \in c_j}}{|V| + \sum_{m=0}^{|V|} n_{t_m \in c_j}} \tag{5}$$

For operation categorization, the same equations (1)-(5) are used, but instead of WS instances (ws_i), we consider operation instances (op_i) and domains c_j are replaced by *ideal*, i.e. semantically described, operations (id_j).

3.3 Building logistic model trees using error correcting codes

The newly vector model representation of the training data set is used as input for building a classifier based on LMT and ECOC. LMT is a relatively new classification algorithm, which combines logistic regression and decision trees. We adopt LMT because it performs efficiently on small and/or noisy data sets. Another advantage of using logistic regression is that it produces as output explicit probability estimates for each class, rather than suggesting a single output class.

As the WS classification task involves a large number of classes, we use the LMT classifier in combination with ECOC for correcting the classification errors. ECOC decompose a multi-class classification problem into many binary classification tasks and then combines the results to decide upon the correct class. It has been shown that the ECOC approach improves the generalization performance of typical classifiers, such as the Naive Bayes, decision tree and SVM, in multi-class problems (Berger, 1999; Ghani, 2002).

To perform categorization of WS operations, we use LMT and ECOC, based on the Bayes' theorem application schema discussed in Section 3.2. We call this composite schema B-LMT-ECOC, and we deploy it for each one of the available domains. To tackle the lack of sufficient training data for operation categorization, we integrated lexicographic matching techniques into our approach, which fine-tune the results of the developed WS categorization framework.

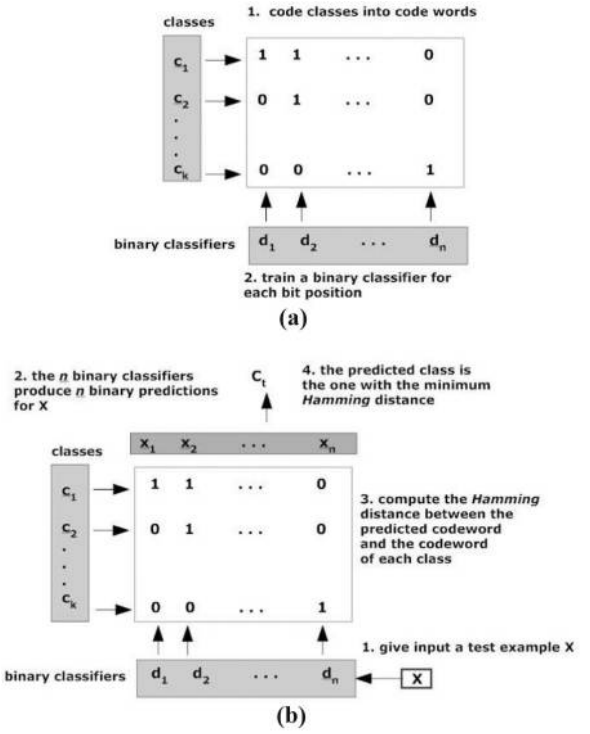
3.3.1 Logistic model tree. LMT consists of a standard decision-tree structure with logistic regression functions at the leaves (Landwehr *et al.*, 2005). It contains N inner nodes and a set of leaves (terminal nodes) T. Let S denote the whole instance space, spanned by all features that are present in data. The tree structure gives a disjoint subdivision of S into regions S_t, and every region is represented by a leaf in the tree, as shown in equation (6):

$$S = \bigcup_{t \in T} S_t,\ S_t \cap S_{t'} = \varnothing, \forall\, t \neq t' \tag{6}$$

Unlike ordinary decision trees, the leaves $t \in T$ have an associated logistic regression function F_t instead of just a class label. The regression function F_t takes into account a subset of all features present in the data. The *LogitBoost* algorithm (Friedman *et al.*, 2000) was adopted for building the logistic regression functions at the tree nodes, which uses the well-known *CART* algorithm for tree pruning (Breiman *et al.*, 1984).

3.3.2 Combining logistic model trees with error correcting codes. ECOC can be efficiently used for enhancing the performance of the most common classifiers (e.g. SVM), in multi-class problems, e.g. (Berger, 1999; Ghani, 2002). Our work leverages the use of ECOC to enhance the performance of the LMT during the WS categorization process.

The ECOC technique consists of two stages, namely, encoding and decoding, as illustrated in Figure 1. Given a set of k classes to be learnt, this technique forms n different bi-partitions and trains n binary classifiers (i.e. "dichotomizers"). As a result, a code word of length n is obtained for each class, where each bit of the code corresponds to the response of a given dichotomizer.



Notes: (a) Encoding; (b) decoding

Figure 1.
The two steps of the ECOC encoding technique (Mavridou *et al.*, 2012)

Therefore, by putting together the code words for all classes, a coding matrix M, of dimension $k \times n$, is formed. The decoding stage involves the classification of an unseen example, based on the value of the output code. Testing an unseen example entails comparison of the output code word from the n classifiers against each class codeword, and the selection of the best match as the class label.

A typical measure of the error-correcting code quality is the minimum Hamming distance between any pair of code words (Hamming, 1950). The Hamming distance between two strings of equal length is the number of positions, for which the corresponding symbols are different. If the minimum Hamming distance is d, then the code can correct at least $(d - 1)/2$ one-bit errors (Witten and Frank, 2005). Thus, if we make $(d - 1)/2$ errors, the nearest code word will still be the correct code word.

Several techniques have been proposed for producing error-correcting codes (Witten and Frank, 2005). The exhaustive approach could be considered as more effective error correcting method compared to the One versus Rest approach. However, when the number of classes k is large (i.e. $k > 7$, according to empirical experiments), it becomes computationally expensive (exponential increment) due to the large length of the code words. As a consequence, too many classifiers have to be built. For this reason, we adopt the exhaustive approach when the number of classes is $k \leq 7$, whereas for $k > 7$, we deploy randomly selected error-correcting codes, which are known to have good error-correcting properties.

3.3.3 Domain categorization process. Figure 2 illustrates the various steps of the WS domain categorization process. For the classification of a test WS, the output code word from the n classifiers is compared to the class code words, and the one with the minimum Hamming distance is selected to be the class label. Before the test WS is passed to the decision tree classifier, it has to be pre-processed to represent it according to the vocabulary V, which was determined at the training phase. After that, the test WS is transformed according to Bayes' rule (in the same way as for the training data) and is given as input to the binary LMT classifiers to predict the class, to which the WS belongs. The leaves of each

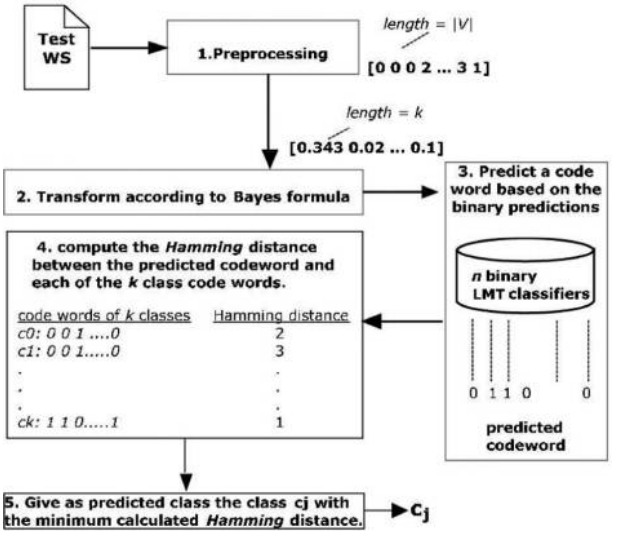


Figure 2.
Classification of an unknown (test) WS

LMT classifier correspond to logistic functions, which produce the class probabilities for the two binary classes "0" and "1".

3.3.4 Operation categorization process. Operation categorization performs matching between a WS operation with an "ideal" operation, i.e. a predefined operation described within the ontology. Thus, this categorization step can be considered as predicting the semantics that characterize a randomly selected WS operation. To this end, we construct a B-LMT-ECOC classifier for each WS domain, after applying the pre-processing procedure and the Bayes' theorem for reducing the feature space dimension, as discussed in Sections 3.1 and 3.2, respectively.

In the case of operation categorization, the training data set consists of operation semantics described in an appropriate ontology, expressed in the Web Ontology Language (OWL). The various steps of the WS operation categorization process are described as follows. Firstly, a pre-processing procedure is conducted to a test WS, for extracting the initial set of features. Then, we represent the WS according to the VSM that results after applying the Bayes' theorem (as described in Section 3.2 for training data). The transformed WS operation is provided as an input to the n binary LMT classifiers, which produce a binary output.

Instead of selecting the class code word with the minimum Hamming distance as the class label, similarly to the domain prediction process, we perform the following adaptation to further assist the classifier to avoid any misclassifications. In particular, we use *WordNet:Similarity* with *Jiang* metric (Pedersen *et al.*, 2004), for comparing lexicographically the name of the test WS operation with the names of the *ideal* operations in the ontology.

Let s be the name of a randomly selected test WS operation and id_i the names of the *ideal* operations in the same domain, $i = 0 \ldots t$. For each *ideal* operation name id_i the *WordNet:Similarity* algorithm gives a matching score wn_i that expresses the degree of similarity between any two words. The B-LMT-ECOC classifier produces a probability for each one of the classes, computed by summing the probabilities that correspond to the binary predictions after normalizing the resulted sum. Thus, given the set of classes $ID = id_0, id_1, \ldots, id_t$ a set of corresponding probabilities $Pr = pr_0, pr_1, \ldots, pr_t$ are calculated. Then, for each id_i, we calculate the final score as: $ts_i = pr_i + wn_i$. This additional lexicographic check improves the categorization results, tackling especially the case of rare instances, i.e. instances that correspond to classes with a few training data.

3.3.5 Input/output parameters categorization. In the case of i/o categorization, an alternative algorithmic approach is proposed, which is based on the Wordnet lexical database. The i/o categorization task is reduced into String comparison process taking into account both i/o data types and their structure.

The purpose of this process is to categorize one by one all i/o parameters from a WS operation into semantically described i/o that belong to *ideal* operations.

The new proposed technique, namely I/O Similarity Matching (IOSM), is based on a one-to-one similarity comparison between the i/o elements, with respect to their names, hierarchy and data type. For this purpose, Wordnet lexical database (Pedersen *et al.*, 2004) has been used, for extracting word Synsets (i.e. cognitive synonyms) and Hypernyms (i.e. semantic annotated synsets with more general meanings). The proposed algorithm for calculating the similarity scores between the *real* and *ideal* i/o parameters is described in Algorithm for the classification of input and output parameters.

3.3.5.1 Algorithm for the classification of input and output parameters.

```
Input-Output-Similarity (ws, WS) : M

01. ws: a web services; WS: ontology
02. M: similarity matrix
03. V_io, V'_io ← null
04. V_io ← parseWSDL (ws)
05. For each v_i ∈ V_io
06.   T_i ← split (vi)
07.   For each t_j ∈ T_i
08.       S_ij ← synsets (tj)
09.       H_i,j ← hypernyms (tj)
10.       C_i,j ← S_i,j ∪ H_i,j
11.   End for
12. End for
13. V'_io ← parseOWL (WS)
14. For each v'_k ∈ V'_io
15.   T'_k ← split (v'_k)
16.   For each t'_l ∈ T'_k
17.       S'_k,j ← synsets (t'_l)
18.       H'_k,l ← hypernyms (t'_l)
19.       C'_k,l ← S'_k,l ∪ H'_k,l
20.   End for
21. End for
22. For each v_i ∈ V_io
23.   For each v'_k ∈ V'_io
24.       G_i,k ← bipartite (C_i,1, C_i,2, ... | C'_k,1, C'_k,2, ...)
25.       For each c_i,j ∈ C_i
26.           For each c'_k,l ∈ C'_k
27.               s_j,l ← WNSimilarity (c_ij, c'_k,l)
28.           End for
29.       End for
30.       L_i,k ← max (s_i,l)
31.       D_i,k ← DatatypeSimilarity (v_i, v'_k)
32.       R_i,k ← StructureSimilarity (v_i, v'_k)
33.       M_i,k ← 0.8*L_i,k+0.1*D_i,k+0.1*R_i,k
34.   End for
35. End for
36. Return M
```

The core functionality that IOSM provides is implemented by the *Input-Output-Similarity* procedure, which is described in pseudo-code in Algorithm for the classification of input and output parameters. Its goal is to produce the similarity matrix S that contains the similarity scores for each pair of *real-ideal* i/o parameters. After calculating S, the Hungarian algorithm (Kuhn, 1955) is applied to determine the best matchmaking between *real-ideal* i/o parameters. The algorithm takes as input an arbitrary WS (denoted by *ws*), whose operations we aim to semantically categorize, and the corresponding WS ontology (denoted by *O*) that describes the class, to which *ws* belongs. As a first step, the algorithm parses the WSDL/WADL description of the particular WS operation we wish to categorize, using function *parseWSDL*, and puts the extracted i/o parameters along with their parent nodes

into vector V_{io}. Then, we split each element $v_i \in V_{io}$, $i = 1, \ldots, |V_{io}|$ into tokens $T_i \doteq \{t_1, t_2, \ldots\}$ and for each $t_j \in T_i, j = 1, \ldots, |T_i|$, we compute its Synset $S_{i,j}$ and Hypernyms $H_{i,j}$ sets, and then their union $C_{ij} \doteq S_{i,j} \cup H_{ij}$. In a similar way, we parse the *ideal* operation, extract the corresponding i/o parameters and insert them into vector V'_{io} along with their parent nodes in the *ideal* operation hierarchy. Again, each element $v'_k \in V'_{io}, k = 1, \ldots, |V'_{io}|$ is split into tokens $T'_k \ddot{:} \{t'_1, t'_2, \ldots\}$. For each $t'_l \in T'_k, l = 1, \ldots, |T'_k|$ the Synsets $S'_{k,l}$ and Hypernyms $H'_{k,l}$ sets are calculated, and then their union is set to $C'_{k,l} \ddot{:} S'_{k,l} \cup H'_{k,l}$. Figure 3 presents the process of extracting the Synsets and Hypernyms of two operation inputs.

The sets $C_{i,j}$ and $C'_{k,l}$ are used for building a bipartite graph, where each entry corresponds to an input or output parameter of the *real* (left) and the *ideal* operation (right). For each pair v_i, v'_k a separate (inner) bipartite graph is built where the left graph elements consist of the sets $C_{i,j}$ and the right ones of the sets $C'_{k,l}$. The scores between sets $C_{i,j}$ and $C'_{k,l}$ are calculated as the maximum score observed after conducting a one-by-one comparison of their element with the Wordnet similarity metrics (Pedersen *et al.*, 2004). Figure 4 shows an example of how the score is calculated. If any of the words in a node consists of multiple tokens, then a new bipartite graph (second level inner graph) is formulated to compute the score between words independently. The comparison between two sets, as in Figure 4, is performed by applying the Wordnet Similarity metrics, forming the bipartite graph that will lead to the optimum matching through the Hungarian algorithm.

The parameters v_i and v'_k are also compared in terms of their data type, forming the matrix $D_{i,k}$. For this purpose the *DatatypeSimilarity* function is used, which returns 1.0 if the data types match exactly, 0.5 if both parameters belong to the same complex data type and 0.0 in any other case. Parameters v_i and v'_k are finally compared in terms of their structure, i.e. if their parent nodes in the operations tree match with each other. The hierarchical tree parents names are stored in two vectors, where they are compared element-by-element by the same algorithm presented, ignoring the steps described in lines 31 and 32 of the algorithm shown in Algorithm for the classification of input and output parameters, and returning the matrix $L_{i,k}$. This is implemented through the *StructureSimilarity* function,

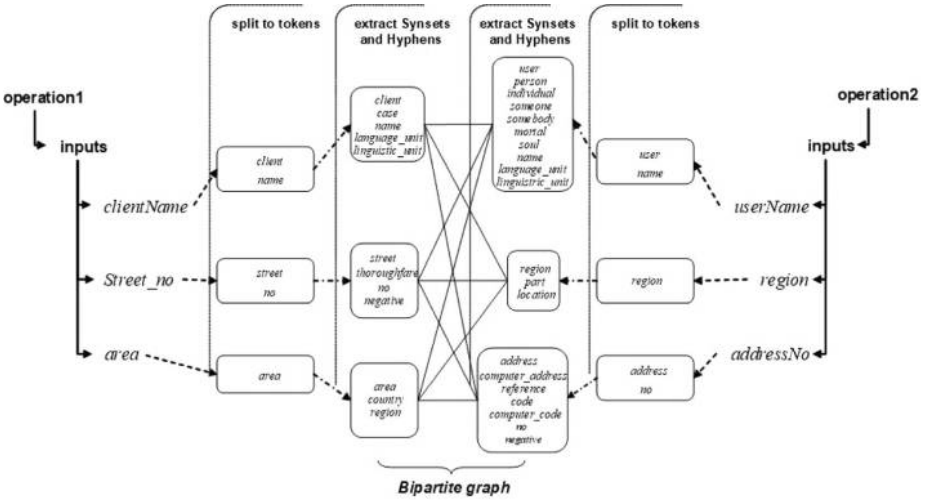


Figure 3.
The preprocessing steps in order to form the bipartite graph for finding the best matches

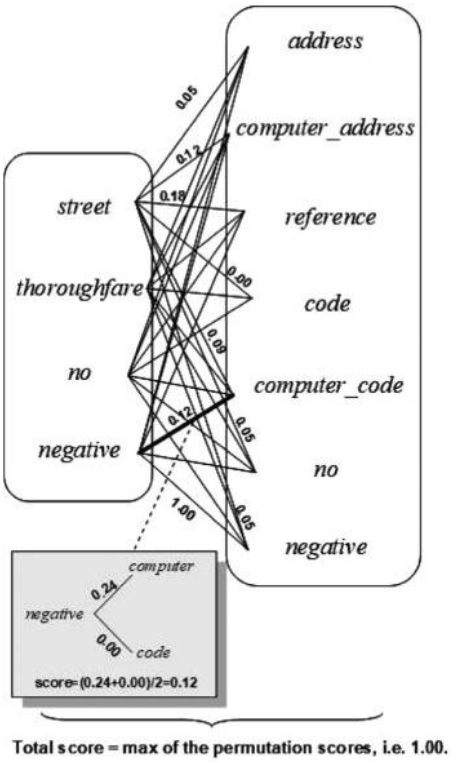


Figure 4.
An example of how the score between two nodes in the bipartite graph is computed

which returns as an output the structure similarity matrix $R_{i,k}$. The final score matrix $M_{i,k}$ is then calculated by equation (7):

$$M_{i,k} = 0.8L_{i,k} + 0.1\,D_{i,k} + 0.1\,R_{i,k} \tag{7}$$

It should be noted that the values of the weights in equation (7) (also shown in line 33 of the algorithm) were selected empirically, as 0.8 for parameter names, 0.1 for their data type and 0.1 for their structure. From empirical experimentation, we have seen that the importance of a term increases by its frequency of appearance in other documents. On the contrary, the structure of complex i/o parameters does not have a significant impact, neither their data type, as in most cases, more generic data types, e.g. String, are used to represent other primitive types, such as integers and floats. These observations resulted in the selected weighting scheme, according to which the parameter name is considered to have more impact with respect to the other attributes, but can be adjusted at will. Finally, the best scores form the final mapping between i/o of the *real* and the *ideal* operations are computed by solving the assignment problem by means of the Hungarian algorithm.

4. Evaluation procedure and results

This section presents the results of the evaluation process that we conducted to assess the accuracy of the proposed semantic categorization framework. To this end, we conducted

two types of evaluation experiments. The first one aims to test our method against different variations of our technique that result after applying different feature selection and extraction approaches, different term weighting schemes and different well-known classification algorithms. The second one aims at benchmarking our technique against a set of known approaches from the relevant literature. In all cases, we use the following metrics: accuracy, precision, recall and F-measure. The objective of the evaluation procedure is to compare our method against others when the subject to be categorized is a WS, an operation and an input or output parameter. We conducted evaluation of B-LMT-ECOC to provide measurable evidence about the improvement of the classification accuracy that is introduced by the application of the dimensionality reduction technique described in Section 3.2.

Specifically, in the domain categorization case, different types of documents, i.e. both WSDL and WADL, are used for synthesizing different data sets. We apply all those data sets to examine the impact of the data on the achieved accuracy evaluation metrics and only when comparing B-LMT-ECOC to its variations. Benchmark against existing third party approaches was based on the OWLS-TC[1] V4.0 data set. In what follows all elements of the experimental procedure are described in detail.

4.1 Data

The original data set that was used for our evaluation procedure contains 249 WSDL and 58 WADL files that were collected from various open WS repositories, such as www.programmableweb.com. Thus, the whole data set counts 307 WS in total[2]. All WS in our collection were manually annotated. Domain-level categorization includes the categorization of the WS into the following six application domains: *Business and Money, Geographic, Communication, Tourism and Leisure, Transport* and *Weather*. Table II shows the six categories, along with the number of all WS in each domain. In addition to the synthetic data set that we compiled as described above, we used the OWLS-TC data set to conduct comparison to existing service categorization techniques.

4.2 Domain categorization

4.2.1 Evaluating various feature selection techniques. In the first experiment, our goal is to examine the impact that the various feature selection techniques have on the accuracy of the service categorization process in various domains, to justify the selection of the optimal one that was used in the design of B-LMT-ECOC. Hence, we implemented a set of variations of B-LMT-ECOC, each one of which implements a different feature selection and extraction technique.

In particular, the following techniques are used: (a) Chi^2 that selects a subset of features based on the chi-squared statistic test with confidence level of $a = 0.05$, (b) *LSA*, i.e. the

Domain name	No. of WSDL files	No. of WADL files
Business and money	96	20
Geographic	55	12
Communication	58	7
Tourism and leisure	13	4
Transportation	10	12
Weather	17	3
Total	249	58

Table II.
The data set used for the evaluation of domain categorization

original feature set is transformed by the application of latent semantic analysis (LSA) where 95 per cent of singular values are used, (c) *IG*, i.e. a subset of features is selected by the use of information gain (IG) where the threshold for the IG is set to 0, (d) *PCA*, i.e. the initial feature set is transformed by the application of principal component analysis (PCA) with 95 per cent of variance covered, and (e) *B-LMT*, which exploits our proposed Bayes' theorem-based feature selection technique (without the use of ECOC). We use those variations as benchmarks to assess the impact that our feature selection, as well as ECOC techniques have on the overall classification accuracy.

The average accuracy, precision, recall and F-measure, obtained by applying the aforementioned techniques on ten random training-test data sets of WSDL files, are presented in Figure 5. The B-LMT-ECOC performance is equivalent but slightly better than the B-LMT on the training-test data set. This is due to the fact that ECOC increases significantly the performance of the classifier for large number of classes that is not our case in the domain categorization task. The rest of the methods are ranked below, i.e. while B-LMT-ECOC and B-LMT achieve accuracy over 90 per cent, the other methods achieve accuracy at about 75 per cent.

4.2.2 Working with RESTful services. For the evaluation of domain categorization of RESTful services, we used the complete WSDL collection described in Table II as our training data set and the 58 WADL files as the test data set. Figure 6 depicts the results achieved for each of the applied techniques, where it is shown that the accuracy reaches the level of 91.67 per cent, whereas precision, recall and F-measure at 76.79, 65.28 and 75.14 per cent, respectively.

The evaluation process was repeated by including different percentage of WSDL and WADL documents in the training set. The results involving WADL files in the training and/or in the test sets, are at the same accuracy level as in the case where only WSDL files were used, i.e. a slight superiority of the B-LMT-ECOC approach. This provides evidence that the classification mechanism is independent of the format of the WS description document.

4.2.3 Evaluating term weighting schemes. The decision on representing the occurrence of features with their frequency was not made arbitrarily. Experiments were made evaluating different term weighting schemes. Specifically, the use of the following term weighting schemes was examined (Salton and Buckley, 1988):

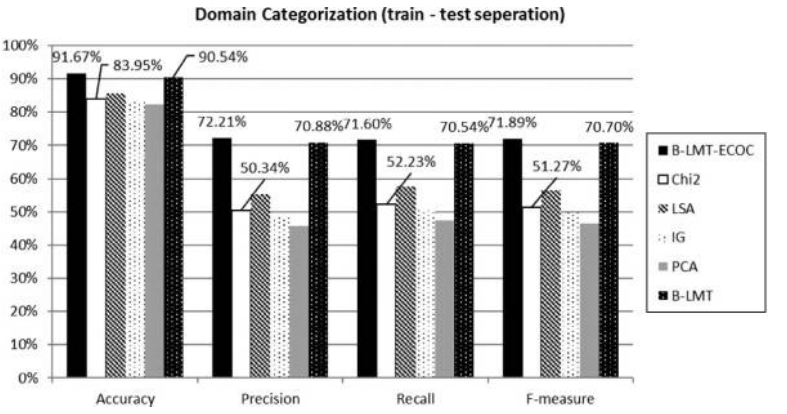


Figure 5.
Domain categorization accuracy, precision, recall and F-measure comparison of B-LMT-ECOC and B-LMT, Chi2, IG, LSA and PCA approaches with ten random training-test data sets

- *TF-IDF*: Instead of using frequency each term occurrence was represented by the following equation which takes also into account the frequency of occurrence of the term in the whole data set:

$$tfidf = tf \cdot idf(t, D), \tag{8}$$

where:

$$idf(t, d) = \log \frac{N}{|\{d \in D : t \in d\}|}. \tag{9}$$

- *Log TF-IDF*: This scheme is a variation of the *TF-IDF* which is described by the following equation:

$$LogTF - IDF = \log(1 + tf) \cdot idf. \tag{10}$$

- *Binary*: A feature is represented by 1 when it occurs and by 0 when it does not occur.

Figure 7 depicts the results achieved for each of the applied schemes where it is shown that the highest accuracy, precision, recall and F-measure are achieved when the Frequency is used for representing the feature occurrence.

4.2.4 Evaluating various well-known classification algorithms. The use of different well-known classification algorithms was examined. Specifically we examined the use of support vector machines, decision trees and logistic regression (Elomaa and Kääriäinen, 2001; Hilbe, 2009; Keerthi *et al.*, 2001). The Weka implementations SMO, REPTree and SGD (logistic regression) were used respectively[3]. We evaluated the aforementioned algorithms against B-LMT-ECOC in the same random train/test data set of the WSDL services of the OWLS-TC collection. As it is depicted from Figure 8, B-LMT-ECOC achieved the highest Accuracy, Precision, Recall and F-measure outperforming the rest of the classifiers.

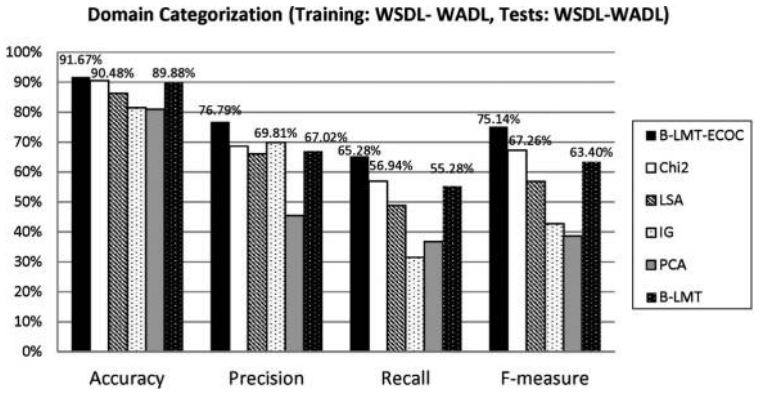


Figure 6.
Comparison of the six domain categorization approaches, when test data contain WSDL and WADL documents, and training data contain WSDL and WADL, as well

4.2.5 Benchmarking B-LMT-ECOC against existing techniques. To evaluate the performance of the B-LMT-ECOC approach in service domain categorization, we have conducted a set of experiments whose purpose is to compare our technique with existing ones from the recent literature. In particular, we compared our approach against NACWS by He *et al.* (2010), the one by Sharma *et al.* (2014) and SemSig by Katakis *et al.* (2009), as well as MWSAF (Oldham *et al.*, 2005; Patil *et al.*, 2004).

Specific care was taken so that the comparison with existing techniques conforms to the same data set. Among the techniques used for comparison, all of them except MWSAF support OWL-S data from the OWLS-TC data set. On the contrary MWSAF only supports WSDL data. To this end, we conducted the following benchmarks.

The first one involves comparison of B-LMT-ECOC with NACWS and SemSig techniques using OWL-S data from the OWLS-TC data set. We reproduced NACWS, as it was not available at the time of this writing; nevertheless, Katakis *et al.* (2009) provided us with the source code of the SemSig method. The results are shown in Figure 9. The achieved accuracy, precision and recall for B-LMT-ECOC are 98.87, 96.27 and 95.87 per cent, respectively, with SemSig being the second best performing technique. In this benchmark,

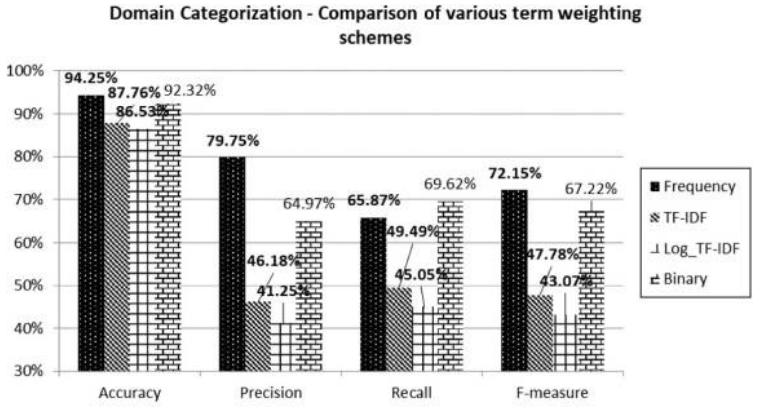


Figure 7.
Comparison of the various term weighting schemes

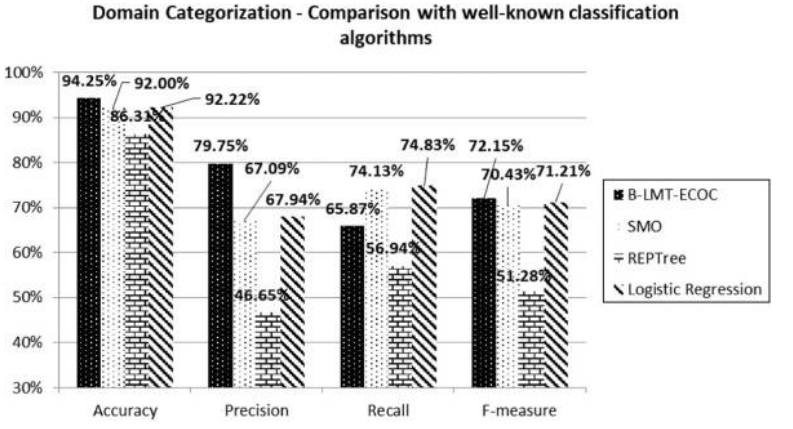


Figure 8.
Comparison with well-known classification algorithms

we failed to include the categorization method by Sharma *et al.* (2014), as it requires an extremely significant computational overhead that renders it practically impossible to be tested over the same data set.

However, for the sake of completeness, we conducted another benchmark taking into account only the domain simulation because it includes only 6 WS, a data set that is manageable by the Sharma technique. Specifically, for the categorization of those WS, it required over 26 h to complete. The average time for one categorization was approximately 4.4 h. Its extreme computational load is justified by the extensive amount of similarity calculations it performs based on a large corpus. For the implementation of the Sharma method, we reproduced it, but we used the Perl script that the authors of Sharma *et al.* (2014) provide for the calculation of the similarity score. The results of this focal benchmark are summarized in Table III, along with the required execution times. As it is shown in Table III, B-LMT-ECOC was the only method that managed to categorize all six WS correctly.

A second benchmark was also conducted that used only WSDL data from the OWLS-TC data set. In this benchmark, we compared B-LMT-ECOC against MWSAF and NACWS. It was not necessary to reproduce MWSAF as it is publicly available[4]. Figure 10 provides a comparison of the three approaches by the means of the four evaluation metrics. Also in this

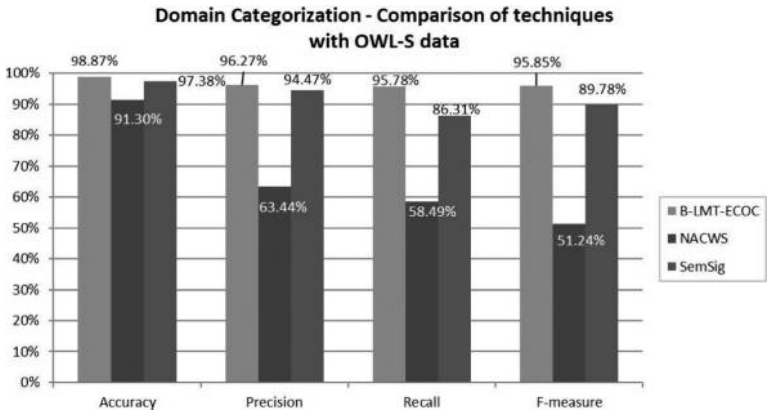


Figure 9.
Comparison of B-LMT-ECOC against NACWS (He *et al.*, 2010) and SemSig (Katakis *et al.*, 2009) service categorization approaches, using OWL-S data from the OWLS-TC data set

Name of WS	Predicted domain (time required for WS classification) (Sharma *et al.*, 2014)	NACWS	B-LMT- ECOC
Unlock door	Travel (6627.64)	Food (78.03)	Simulation (0.0048)
Red light to off	Travel (13585.78)	Food (139.92)	Simulation (0.0037)
Red light to on	Travel (22826.70)	Food (139.52)	Simulation (0.0039)
Green light to off	Travel (12146.71)	Food (138.55)	Simulation (0.0038)
Flip down slider	Travel (32487.05)	Simulation (254.55)	Simulation (0.0048)
Switch off MesseModul	Travel (7104.01)	Simulation (254.55)	Simulation (0.0036)

Table III.
Results of WS classification for the domain "simulation" of the OWL-TC V4.0 data set and time in seconds required by He *et al.* (2010) and Sharma *et al.* (2014) and B-LMT-ECOC

benchmark, B-LMT-ECOC exhibits the best performance, whereas NACWS shows the second best performance for all metrics.

4.3 Operation categorization

To benchmark the B-LMT-ECOC approach in operation categorization, a set of WS operations from the collection D was annotated. Table IV outlines the annotated data set, presenting the number of WS operations that correspond to the predefined application domains. For all data in each domain, we benchmarked the B-LMT-ECOC operation categorization mechanism and all the subsequent methods by applying the leave-one-out cross validation technique for selecting the training and test data sets (see Section 3.1). The benchmark included the variations of the B-LMT-ECOC approach when applying the different feature selection and extraction techniques and MWSAF which supports the categorization of operations of WS.

In Figure 11, the accuracy, precision, recall and F-measure metrics are presented for the operation level classification. The proposed B-LMT-ECOC technique achieves better performance with respect to the other benchmarking methods for all the evaluation metrics. Comparing with B-LMT, the difference is considerable, proving the evidence that the ECOC approach enhances the classifier capability as the number of classes increases.

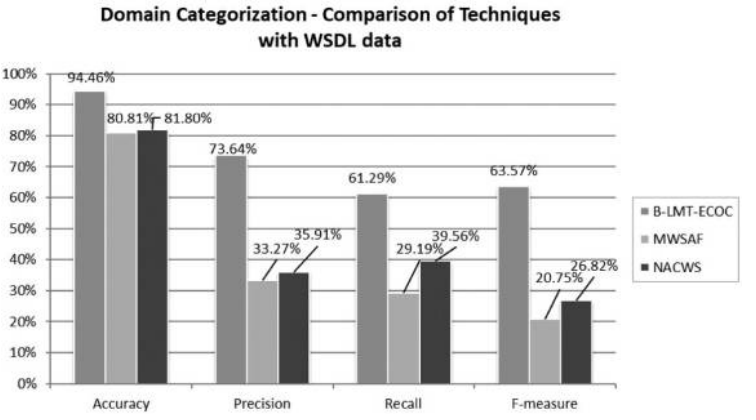


Figure 10.
Comparison of B-LMT-ECOC against MWSAF (Patil *et al.*, 2004) and NACWS (He *et al.*, 2010) service categorization approaches, using WSDL data from the OWLS-TC data set

Domain name	No. of WS operations
Business and money	196
Geographic	110
Communication	89
Tourism and leisure	29
Transportation	23
Weather	48
Total	495

Table IV.
Data set used for the evaluation of operation categorization

4.4 Inputs-outputs categorization

The last part of the evaluation process deals with benchmarking the performance of the i/o categorization approach described in Section 3.3.5, by means of its matching accuracy, precision, recall and F-measure. The purpose of the parameter level categorization is to map all i/o parameters of a WS operation to the semantically described i/o parameters, i.e. the *ideal* operation i/o parameters, which are defined by the WS ontology. The *ideal* operation that is used in this context is the one that matches the *real* operation, to which the tested i/o operation parameters belong.

The process uses 116 i/o parameters from 9 *real* WS operations. Table V summarizes the obtained accuracy after applying various weight configurations, which appears in line 33 of the i/o similarity algorithm (Figure 4).

To evaluate the overall accuracy of IOSM, we conducted comparison to MWSAF and the Lexicon-based alignment approach (Canturk and Senkul, 2011). For the OWLS-TC data set, both approaches achieved 100 per cent accuracy, as the WSDL i/o are almost identical with the OWL-S concepts matched (for example for input _ *MAXPRICE* the respective OWL-S concept is *MaxPrice*. To this end, we used our synthetic data set that was used in Section 4.2.1.

The results of this comparison are shown in Figure 12. In particular, our IOSM method achieved marginally better accuracy than Lexicon. Specifically, the average accuracy for IOSM and Lexicon was 99.88 and 99.74 per cent, respectively, with an average F-measure of 98.70 and 90.27 per cent, respectively. In this benchmark, IOSM used the weight set (0.8, 0.1, 0.1), for 15 arbitrarily selected i/o parameters.

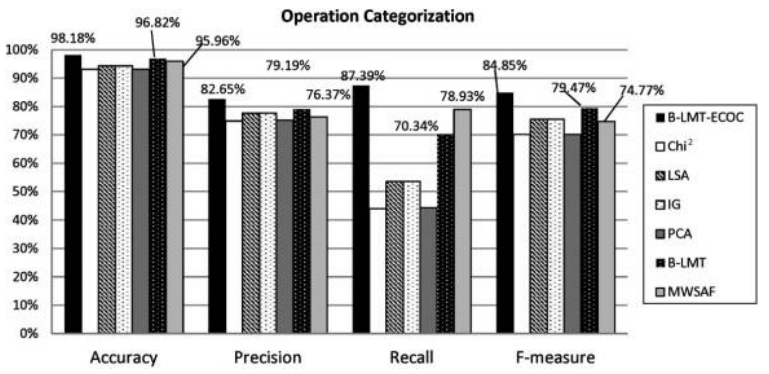


Figure 11. Operation categorization accuracy, precision, recall and F-measure comparison for our approaches B-LMT-ECOC and B-LMT, Chi^2, IG, LSA, PCA and MWSAF approaches with leave-one-out evaluation

$L_{i,k}$	$D_{i,k}$	$R_{i,k}$	IOSM
0.5	0.25	0.25	0.8733
0.8	0.1	0.1	0.9827
0.9	0.05	0.05	0.9827
1.0	0.0	0.0	0.9827

Table V. I/O Categorization results (accuracy) for various values of weight parameters

5. Conclusions

This paper introduced an efficient three-level categorization framework for WS, which can be used for automatic semantic annotation of WS elements. The proposed technique extends the work of Giannoutakis *et al.* (2015) and Mavridou *et al.* (2012) and presents an advanced three-level categorization scheme which categorizes semantically a non-semantic WS and its elements. It deals with the multiclass nature of the classification problem by deploying ECOC (Hamming, 1950) and decomposing the multi-class classification problem into many binary classification ones and finally combining all results to decide upon the correct class. Based on empirical evaluation, our work ascertains that the use of the ECOC increases the classification accuracy of the WS mechanism, especially for the operation categorization task, where the number of classes is too large, e.g. compared to the domain categorization task. To decrease the dimension of the feature space, as a big number of features may have a negative impact on the classification performance, it applies a feature selection process on top of existing WS documents based on Bayes theorem (Mavridou *et al.*, 2012). Finally, it deals with the categorization of i/o WS parameters by the use of bipartite graphs for semantic similarity matching along with Wordnet dictionary and the Hungarian algorithm (Kuhn, 1955). The advancement that is introduced in this respect is that our approach combines in an innovative fashion lexicographic, data type and structure-related characteristics to calculate the semantic similarities of WS i/o parameters. The three levels of categorization (domain, operation and i/o) are integrated into a single framework for automatic WS categorization that comprises the main contribution of this paper.

The proposed WS categorization framework was tested with the OWLS-TC v4.0, as well as a synthetic data set with a systematic evaluation procedure that enables comparison with well-known approaches, using the OWLS-TC data set, as well as several variations of our WS categorization framework. After conducting exhaustive evaluation experiments on the above cases, we measured categorization efficiency in terms of accuracy, precision, recall and F-measure. The presented WS categorization framework outperformed the other benchmark techniques, which comprise different variations of it but also third-party implementations.

Future research will focus on applying our proposed mechanism for the implementation of a semantic annotation framework that will exploit the results of the presented WS categorization framework to produce semantic WS in OWL-S.

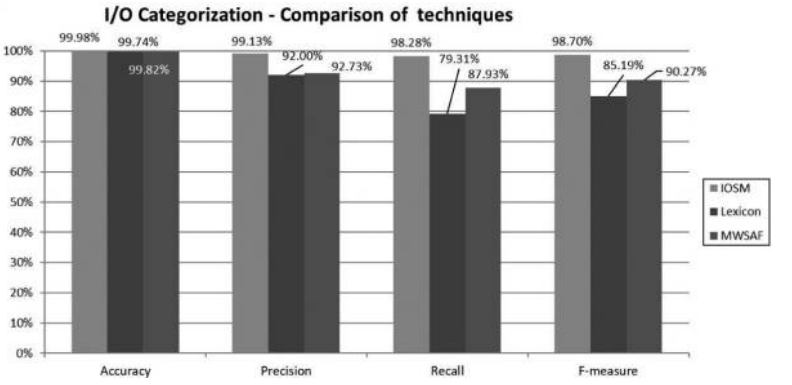


Figure 12.
Accuracy, precision, recall and F-measure of the different i/o categorization approaches

Notes

1. http://projects.semwebcentral.org/projects/owls-tc/
2. The data set used in this paper is available for download at: www.iti.gr/~diok/WSCategorizationData.zip
3. www.cs.waikato.ac.nz/ml/weka/
4. http://lsdis.cs.uga.edu/projects/meteor-s/downloads/mwsaf

References

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A. and Verma, K. (2005), 'Web service semantics-WSDL-S', *A Joint UGA-IBM Technical Note, Version 1*, available at: www.w3.org/Submission/WSDL-S/

Alfazi, A., Sheng, Q.Z., Qin, Y. and Noor, T.H. (2015), "Ontology-based automatic cloud service categorization for enhancing cloud service discovery", *Enterprise Distributed Object Computing Conference (EDOC), 2015 IEEE 19th International*, IEEE, Washington, DC, pp. 151-158.

Berger, A. (1999), "Error-correcting output coding for text classification", *IJCAI-99: Workshop on Machine Learning for Information Filtering*.

Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984), *Classification and Regression Trees*, Wadsworth and Brooks, Monterey, CA.

Bruno, M., Canfora, G., Penta, D.M. and Scognamiglio, R. (2005), "An approach to support web service classification and annotation, in e-Technology, e-Commerce and e-Service, 2005", '*EEE 05 Proceedings. The 2005 IEEE International Conference*', IEEE, Washington, DC, pp. 138-143.

Canturk, D. and Senkul, P. (2011), "Semantic annotation of web services with lexicon-based alignment", *Services (SERVICES), 2011 IEEE World Congress*, IEEE, Washington, DC, pp. 355-362.

Chen, L., Wang, Y., Yu, Q., Zheng, Z. and Wu, J. (2013), "Wt-lda: user tagging augmented lda for web service clustering", in Basu, S., Pautasso, C., Zhang, L. and Fu, X. (Eds), *Service-Oriented Computing*, Lecture Notes in Computer Science, Springer, Berlin Heidelberg, Vol. 8274, pp. 162-176.

Chinnici, R., Moreau, J.J., Ryman, A. and Weerawarana, S. (2007), "Web services description language (wsdl) version 2.0 part 1: core language", World Wide Web Consortium, Recommendation REC-wsdl20-20070626, available at: www.w3.org/TR/2007/REC-wsdl20-20070626/

Elomaa, T. and Kääriäinen, M. (2001), "An analysis of reduced error pruning", *Journal of Artificial Intelligence Research*, Vol. 15 No. 1, pp. 163-187, available at: http://dl.acm.org/citation.cfm?id=1622845.1622850

Eyheramendy, S., Lewis, D.D. and Madigan, D. (2003), "On the naive bayes model for text categorization", in Bishop, C.M. and Frey, B. J (Eds), *Proceedings of the Nineth International Workshop on Artificial Intelligence and Statistics*, pp. 332-339.

Fang, L., Wang, L., Li, M., Zhao, J., Zou, Y. and Shao, L. (2012), "Towards automatic tagging for web services", *Web Services (ICWS), 2012 IEEE 19th International Conference*, IEEE, Washington, DC, pp. 528-535.

Friedman, N., Geiger, D. and Goldszmidt, M. (1997), "Bayesian network classifiers", *Machine Learning*, Vol. 29 No. 2, pp. 131-163.

Friedman, J., Hastie, T. and Tibshirani, R. (2000), "Additive logistic regression: a statistical view of boosting", *The Annals of Statistics*, Vol. 28 No. 2, pp. 337-407.

Ghani, R. (2002), "Combining labeled and unlabeled data for multiclass text categorization", *Proceedings of the Nineteenth International Conference on Machine Learning, ICML '02, Morgan Kaufmann Publishers, San Francisco, CA*, pp. 187-194.

Giannoutakis, K.M., Kehagias, D.D. and Tzovaras, D. (2015), "A three-level semantic categorization scheme of web services", *2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA)*, IEEE, Washington, DC, pp. 73-77.

Hadley, M.J. (2006), Web application description language (wadl), Technical report, Sun Microsystems, Mountain View, CA.

Hamming, R. (1950), "Error detecting and error correcting codes", *Bell System Technical Journal*, Vol. 26 No. 2, pp. 147-160.

Hao, Y., Zhang, Y. and Cao, J. (2010), "Web services discovery and rank: an information retrieval approach", *Future Generation Computer Systems*, Vol. 26 No. 8, pp. 1053-1062, available at: www.sciencedirect.com/science/article/pii/S0167739X10000762

Hao, Y. and Zhang, Y. (2007), "Web services discovery based on schema matching", in Dobbie, G., (Ed.), *Thirtieth Australasian Computer Science Conference (ACSC2007)*, *CRPIT, ACS*, *Ballarat*, Vol. 62, pp. 107-113.

Hasan, M.H., Jaafar, J. and Hassan, M.F. (2014), "Fuzzy-based clustering of web services' quality of service: a review", *JCM*, Vol. 9 No. 1, pp. 81-90.

He, J., Li, P., Hu, X. and Wu, X. (2010), "A new automatic categorization algorithm for web services", *Granular Computing (GrC), 2010 IEEE International Conference*, IEEE, Washington, DC, pp. 200-205.

Heß, A., Johnston, E. and Kushmerick, N. (2004), "Assam: a tool for semi-automatically annotating semantic web services", *In Intl. Semantic Web Conf. (ISWC)*, Springer, Berlin, Heidelberg, pp. 320-334.

Hilbe, J.M. (2009), *Logistic Regression Models*, CRC press.

Isa, D., Lee, L., Kallimani, V. and RajKumar, R. (2008), "Text document preprocessing with the bayes formula for classification using the support vector machine", *Knowledge and Data Engineering, IEEE Transactions on*, Vol. 20 No. 9, pp. 1264-1272.

Katakis, I., Meditskos, G., Tsoumakas, G., Bassiliades, N. and Vlahavas, (2009), "On the combination of textual and semantic descriptions for automated semantic web service classification", *in* Iliadis, Maglogiann, Tsoumakasis, Vlahavas and Bramer Eds, Artificial Intelligence Applications and Innovations III, *IFIP International Federation for Information Processing*, *Springer*, *US*, Vol. 296, pp. 95-104.

Keerthi, S.S., Shevade, S.K., Bhattacharyya, C. and Murthy, K.R.K. (2001), "Improvements to platt's smo algorithm for svm classifier design", *Neural Comput*, Vol. 13 No. 3, pp. 637-649, available at: http://dx.doi.org/10.1162/089976601300014493

Khorasgani, R., Stroulia, E. and Zaiane, O. (2011), "Web service matching for restful web services", *Web Systems Evolution (WSE), 2011 13th IEEE International Symposium*, IEEE, Washington, DC, pp. 115-124.

Kopecky, J., Gomadam, K. and Vitvar, T. (2008), "hrests: an html microformat for describing restful web services", *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08. IEEE/WIC/ACM International Conference*, IEEE, Washington, DC, Vol. 1, pp. 619-625.

Kuhn, H.W. (1955), "The hungarian method for the assignment problem", *Naval Research Logistics Quarterly*, Vol. 2 Nos 1/2, pp. 83-97.

Landwehr, N., Hall, M. and Frank, E. (2005), "Logistic model trees", *Machine Learning*, Vol. 59 Nos 1/2, pp. 161-205.

Li, M. and Yang, Y. (2012), "Efficient clustering index for semantic web service based on user preference", *Computer Science and Information Processing (CSIP), 2012 International Conference*, IEEE, Washington, DC, pp. 291-294.

Liang, Q., Li, P., Hung, P. and Wu, X. (2009), "Clustering web services for automatic categorization", *Services Computing, 2009. SCC '09. IEEE International Conference*, IEEE, Washington, DC, pp. 380-387.

Mccallum, A. and Nigam, K. (1998), "A comparison of event models for naive bayes text classification", *AAAI-98 Workshop on Learning for Text Categorization*.

Maleshkova, M., Pedrinaci, C. and Domingue, J. (2009), "Supporting the creation of semantic restful service descriptions", *8th International Semantic Web Conference (ISWC 2009)*, CEUR-WS.

Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., McGuinness, D., Sirin, E. and Srinivasan, N. (2007), "Bringing semantics to web services with owl-s", *World Wide Web*, Vol. 10 No. 3, pp. 243-277.

Mavridou, E., Hassapis, G., Kehagias, D.D. and Tzovaras, D. (2012), "Semantic categorization of web services based on feature space transformation", *Informatics (PCI), 2012 16th Panhellenic Conference*, IEEE, Washington, DC, pp. 162-167.

Mustafa, S.A. and Swamy, K.Y.S. (2015), "Web service classification using multi-layer perceptron optimized with tabu search", *Advance Computing Conference (IACC), 2015 IEEE International*, IEEE, Washington, DC, pp. 290-294.

Oldham, N., Thomas, C., Sheth, A. and Verma, K. (2005), "Meteor-s web service annotation framework with machine learning classification", in J. Cardoso and A. Sheth, (Eds), *Semantic Web Services and Web Process Composition*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, Vol. 3387, pp. 137-146.

Patil, A., A., Oundhakar, S., A., Sheth, A., P., and and Verma, K. (2004), "Meteor-s web service annotation framework", *Proceedings of the 13th International Conference on World Wide Web, WWW '04, ACM, New York, NY*, pp. 553-562.

Pedersen, T., Patwardhan, S. and Michelizzi, J. (2004), "Wordnet::similarity: measuring the relatedness of concepts", *Demonstration Papers at HLT-NAACL 2004', HLT-NAACL–Demonstrations 04, Association for Computational Linguistics, Stroudsburg, PA*, pp. 38-41.

Porter, M. (1980), "An algorithm for suffix stripping", *Program: Electronic Library and Information Systems*, Vol. 14 No. 3, pp. 130-137.

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. and Fensel, D. (2005), "Web service modeling ontology", *Appl. Ontol*, Vol. 1 No. 1, pp. 77-106.

Salton, G. and Buckley, C. (1988), "Term-weighting approaches in automatic text retrieval", *Inf. Process. Manage*, Vol. 24 No. 5, pp. 513-523, available at: http://dx.doi.org/10.1016/0306-4573(88)90021-0

Sammut, C. and Webb, G.I. (2011), *Encyclopedia of Machine Learning*, Springer Science & Business Media.

Sebastiani, F. (2002), "Machine learning in automated text categorization", *ACM Comput. Surv*, Vol. 34 No. 1, pp. 1-47.

Sellami, S. and Boucelma, O. (2011), "Web services discovery and composition: a schema matching approach", *Web Services (ICWS), 2011 IEEE International Conference*, pp. 706-707.

Sharma, S., Lather, J. and Dave, M. (2014), "Semantic approach for classification of web services using unsupervised normalized similarity measure", *Journal of Emerging Technologies in Web Intelligence*, Vol. 6 No. 3.

Sheth, A., Gomadam, K. and Lathem, J. (2007), "Sa-rest: semantically interoperable and easier-to-use services and mashups", *Internet Computing, IEEE*, Vol. 11 No. 6, pp. 91-94.

Verma, K. and Sheth, A. (2007), "Semantically annotating a web service", *IEEE Internet Computing*, Vol. 11 No. 2, pp. 83-85.

Vitvar, T., Kopeckỳ, J., Viskova, J. and Fensel, D. (2008), "Wsmo-lite annotations for web services", in Bechhofer, S., Hauswirth, M., Hoffmann, J. and Koubarakis, M. (Eds), *The Semantic Web: Research and Applications*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 674-689.

Wang, H., Shi, Y., Zhou, X., Zhou, Q., Shao, S. and Bouguettaya, A. (2010), "Web service classification using support vector machine", *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference*, IEEE, Washington, DC, Vol. 1, pp. 3-6.

Wang, J., Zhang, J., Hung, P., Li, Z., Liu, J. and He, K. (2011), "Leveraging fragmental semantic data to enhance services discovery", *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference*, IEEE, Washington, DC, pp. 687-694.

Witten, I.H. and Frank, E. (2005), *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*, Morgan Kaufmann Publishers, San Francisco, CA.

Yang, Y. and Pedersen, J.O. (1997), "A comparative study on feature selection in text categorization", *Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97, Morgan Kaufmann Publishers Inc., San Francisco, CA*, pp. 412-420.

Corresponding author
Konstantinos M. Giannoutakis can be contacted at: kgiannou@iti.gr