An Enhanced Approach for Web Services Clustering using Supervised Machine Learning Techniques

Abdelmoniem Helmy, Mervat H. Geith

Abstract—Automatic document classification provides techniques that may improve and support web service clustering. As the number of services increases, the cost of classifying services through manual work increases. In this research, we presented an enhanced approach for service clustering that combines text mining and machine learning technology. The method only uses text description of each service so that it can classify different types of services, such as WSDL Web Service, RESTful Web Service. This approach provides better performance in terms of service discovery efficiency and effectiveness. In this paper, we introduced new features, named referenced ontology, that can be extracted from WSDL documents and integrated to other features to get better classification accuracy. This paper proposes a text mining approach to automatically classify services to specific domains and identify key concepts inside service textual documentation. This approach is validated on a dataset of 1088 web services categorized into 9 fields yielding accuracy up to 91 %. Our approach utilizes the supervised machine learning techniques such as Decision Trees, Naïve Bays, and Deep Learning classification methods. A comparison between these techniques are made regarding the result accuracy and the computation cost.

Index Terms—Deep Learning Classification, Decision Tree Classification, Naïve Bayes Classification, Supervised Machine Learning Classification, Text Mining, Web Services Classification, Web Services Clustering, Web Services Discovery.

1. INTRODUCTION

Web service discovery is becoming difficult task because of increasing Web services available on the Internet. As seeking for efficient web service discovery is main challenge for researchers, research in cluster analysis of web services has recently gained much attention. This is due to the popularity of web services and the potential benefits that can be achieved from cluster analysis of web services like reducing the search space of a service search task. The classification process assists the search engine in identifying the functionality of the web service and better matching web services with users’ requests. In this research, we identified four keys features that can be extracted from WSDL or RESTful documents and integrated to cluster web services into functionality-based groups. These features are WSDL content, types, referenced ontology, and web service name. Integrating these features could create a more accurate representation of the functionality of a web service. Our approach adopts both text classification and element classification. In text classification, the proposed approach extracts important information from web service description documents.

2. RELATED WORK

Automatic text classification is becoming an important research topic. In general, automatic text classification plays a vital role in text summarization, question answering and information extraction. The existing research on web service classification has primarily focused on two aspects: text classification and element classification. In text classification, the textual description in a WSDL are mapped into a vector by text mining methods and then applying machine learning technique. Element classification is based on every element in a WSDL. The most widely employed approaches are similarity based, including (1) semantic based and (2) non-semantic based[1]. Ontology is utilized to compute the semantic similarity between Web services in many studies. Specifically, Cristina et al. [2] propose to use an ant-based method to cluster Web services based on semantic similarity. In this research, we focus on the clustering of non-semantic Web services, as such services are more popular and more widely supported by the industry circle.

For the calculation of non-semantic similarity between Web services, WSDL-based approaches are the most representative work [12],[13], [14]. Liu et al. [12],[13] propose to extract four features, i.e., content, context, host name, and service name, from the WSDL document to cluster Web services. Khalid et al. [14] also propose to extract features from WSDL documents to cluster Web services. Different from the work [12], [13], Khalid extracts content, types, messages, ports, and service name from WSDL documents. The work in [19] proposes a text mining based approach for web service classification. classification results prove that instead of using different
WSDL features separately and assigning them different weight, careful selection of suitable attributes from WSDL document and using them collectively as a single feature can improve classification accuracy.

Zhuang et al. [5] presented an algorithm to compute the similarity of two web services using WordNet. Their approach uses the information given in the WSDL files. They do manual pre-processing of the WSDL files to remove abbreviations. This approach does not yield high precision. This is because when the name of an element is consisted of multiple words; those words are considered a unit when the system calculates the similarity score between this element and another. In our approach, we separated compound words into single words using WordNet as a pre-processing step before classification process.

Different examination applied semi-supervised classification in web services composition [23]. A semi-supervised classification algorithm uses a large amount of unlabeled data, supporting the supervised learning process in order to improve classification results. Semi-supervised learning can be divided into semi-supervised classification and unsupervised clustering. The particular semi-supervised clustering algorithm used in this approach clusters data based on tags and constraints.

In addition, other relevant work is presented in [19], as they expose a text mining approach to web services classification by identifying key concepts in textual documentation services, but only in a specific domain. The work presents a text mining approach to the automation classification of web services in specific domains and identification of concepts within the textual documentation of the services. His Approach was validated with a set of 600 web services categorized into 8 classes.

Table 1 gives in chronological order a summary for previous research work on the area of web services clustering and classification. The summary characterizes each work according to five dimensions; features extracted, clustering technique, test data set, contribution, and limitation or recommendation by the author(s).

<table>
<thead>
<tr>
<th>Work</th>
<th>Extracted Features</th>
<th>Technique</th>
<th>Data set</th>
<th>Contribution</th>
<th>Limitation / Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang, &amp; Pan, 2008</td>
<td>WSDL description, and input/output for operations. Several traditional text mining algorithms like tf-idf and try different kernels like polynomial kernel, RBF kernel and string kernel</td>
<td>Several classifiers. Weka 3.4.12 classification tool. Cross-validation is used to estimate the accuracy of the algorithms.</td>
<td>extract information about 500 web services published on <a href="http://www.xmethods.net">http://www.xmethods.net</a></td>
<td>Several classifiers are used and compared. They are SMO, Naive Bayes, Bagging/DecisionDump, AdaBoostM1/DecisionDump, Random Forest</td>
<td>build up an automatic composing system, to build a new web service based on the accurate classification information</td>
</tr>
<tr>
<td>Platzer et al., 2009</td>
<td>content, context, host name, and name. Porter stemmer Poisson distribution</td>
<td>Tree-Traversing Ant (TTA) algorithm NGD as similarity method</td>
<td>400 online Web services - WSDL</td>
<td>Discover functional related services</td>
<td>refine the process of features mining, provide mathematical justifications for word semantic relatedness</td>
</tr>
<tr>
<td>Liu &amp; Wong, 2009</td>
<td>Type, port types, messages, url, comments, Semantic description, location information, QoS.</td>
<td>Modified k-means like Hierarchical Modified cosine similarity called Multidimensional Angle</td>
<td>Manually constructed a small test set of 22 wsdl files</td>
<td>efficient and scalable clustering implementation</td>
<td>results are only based on datasets from Google</td>
</tr>
<tr>
<td>Elgazzar et al., 2010</td>
<td>Content, types, messages, ports, name. Porter stemmer Poisson dist. for function word removal</td>
<td>centroid-based clustering (Quality Threshold) algorithm k-means &amp; NGD for content words recognition</td>
<td>69 published wsdl files</td>
<td>feature extraction for non-semantic web service descriptions higher precision and recall</td>
<td>More processing time. Improves features integration by choosing optimized weights for each feature using a linear programming approach.</td>
</tr>
</tbody>
</table>
3. PROPOSED APPROACH

Framework

In the research community, the main approach for text classification is based on supervised machine learning techniques. These techniques require an initial data set of classified documents to build a classifier. In a Naïve Bayesian classifier, all terms in text are equally important but we adopted Bai and Li [3] suggestion that terms in each title are more significant, so the precision of the classification results can be improved. A web service operation names usually contain some terms describing their functionality and also, WSDL allows embedding natural language descriptions. The lexical algorithms remove stop words from those descriptions; find synonyms using lexical databases like WordNet [4] and compute similarity coefficients.

In our approach, the Web services are converted into a standard vector format through the Web service description document. With the help of WordNet, a semantic analysis is conducted to reduce the dimension of the term vector and to make semantic expansion. The first step is to process web services data from WSDL files using tokenization by non-letters and English stop words removal with vector creation using Term Occurrences. The Term Occurrence will guide us how to prune general xml words that appear in all documents because they are not descriptive words. although there are some common words that has many occurrences but we detect that these words are common to specific domains rather than all documents. Although these words may lead to misclassification of services, but they shall be kept because they are descriptive words. By specifying pruning rule percentual for words that occur above 90% of the documents the pruning rule removed common words that has many occurrences but we detect that they are not descriptive words. although there are some common words that appears in all documents because they are not descriptive words. although there are some common words that has many occurrences but we detect that these words are common to specific domains rather than all documents. Although these words may lead to misclassification of services, but they shall be kept because they are descriptive words. By specifying pruning rule percentual for words that occur above 90% of the documents the pruning rule removed xml common words and leave words common to specific domains.

By specifying pruning rule percentual for words that occur above 90% of the documents the pruning rule removed xml common words and leave words common to specific domains. although there are some common words that has many occurrences but we detect that these words are common to specific domains rather than all documents. Although these words may lead to misclassification of services, but they shall be kept because they are descriptive words. By specifying pruning rule percentual for words that occur above 90% of the documents the pruning rule removed xml common words and leave words common to specific domains.

Text Pre-processing steps are applied to the text to extract accurate and consistent information. Tokenization splits a stream of text into words, symbols and phrases called tokens. Although text is stored in machine-readable format, meaningless characters must be eliminated. The approach proposes the use of word splitter as a new pre-processing step that splits concatenated words based on their case. Then, stop words and function words are filtered from the textual information. In this process, we found some concatenated words that need to be tokenized. The challenge her is that these

<table>
<thead>
<tr>
<th>Work</th>
<th>Extracted Features</th>
<th>Technique</th>
<th>Data set</th>
<th>Contribution</th>
<th>Limitation / Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen et al., 2011 [15] and Chen et al., 2013 [16]</td>
<td>WSDL(Content, types, messages, ports, name) + Tags</td>
<td>k-means algorithm</td>
<td>200 wsdl file from 15,968 real web services Seekda!</td>
<td>explores the knowledge behind WSDL documents and user-contributed tagging data to cluster Web services.</td>
<td>Tag Recommender for web services with few tags</td>
</tr>
<tr>
<td>Vijayan &amp; Balasundaram, 2013 [17]</td>
<td>WSDL(Content, name) + Porter stemmer Poisson distribution and overestimation factor for function word removal</td>
<td>k-means &amp; NGD for content words recognition</td>
<td>N/A</td>
<td>Better performance in terms of time complexity</td>
<td>extend this work, by including additional features such as context to support context-aware pervasive web services and cloud based web services.</td>
</tr>
<tr>
<td>Cong &amp; Gil, 2013 [18]</td>
<td>Semantic Inputs/Outputs: owl-s Syntactic Inputs/Outputs: wsdl Tags</td>
<td>Hierarchal / complete linkage – A Naïve algorithm</td>
<td>(OWLS-TCv4) 1083 web services described in OWL-S and 42 queries</td>
<td>Reduction of processing time from conventional O(n) to O(logn).</td>
<td>Low drop in precision</td>
</tr>
<tr>
<td>Wu et al., 2014 [19]</td>
<td>WSDL(Content, types, messages, ports, name) + Tags + Poisson distribution and overestimation factor for function word removal</td>
<td>k-means NGD as similarity method</td>
<td>crawl 15,968 real web services from the search engine Seekda!</td>
<td>tag mining method choose terms with top TF-IDF as the initial tags</td>
<td>Propose hybrid Web service tagrecommendation strategy to handle the clustering performance limitation caused by uneven tag distribution and noisy tags.</td>
</tr>
<tr>
<td>Nisa &amp; Qamar, 2015 [19]</td>
<td>service name, service documentation, WSDL messages, WSDL, ports and WSDL schema yielding accuracy up to 90%</td>
<td>The supervised machine learning algorithm Maximum entropy</td>
<td>600 web services from web service publisher websites</td>
<td>automatically classify services to specific domains and (2) identify key concepts inside service textual documentation.</td>
<td>use maximum a posteriori estimation for the exponential model, instead of maximum likelihood estimation</td>
</tr>
</tbody>
</table>
words do not follow camel case convention, hence we tokenized them according to word list from WordNet. Tokenizing these compound words gives two advantages; first, it allows us to get more semantically descriptive words about the service, and secondly, it allows us to reduce the vector dimension in the next step. Also, we have to label each example in the example set with service name.

### 3.1 Features Extraction

The various clustering algorithms that are used in this field use the vector space model (see [6]) to represent each document. A widely-used refinement to this model is to weight each term based on its inverse document frequency (IDF) in the document collection. Then the function words and content words are differentiated by calculating the overestimation factor for all the words in each vector. The content words for the Web services then discovered by clustering each word vector into two groups using the k-means clustering algorithm, in which NGD is used as a featureless similarity measure between words. We used Python as a scripting language for calculating the NGD and similarity factor.

Fig.1 shows the four features extracted from each service WSDL file; one content feature and three element features. The content feature generated from all descriptive words extracted after text pre-processing. The elements features include complex types, service name, and referenced ontologies.

![Fig.1. WSDL Features Extraction Process](http://www.ijser.org)

#### 3.2.1 Feature 1: Content

Extracting features from the WSDL document took place in several steps as shown in Fig. 1. Parsing, Tag removal, Content word recognition. A vector of meaningful content words for the given Web service $S_i$ can be extracted by means of processing the WSDL document contents

**Parsing WSDL:** A vector of tokens $T_i$ can be produced by parsing the given WSDL document contents with respect to white spaces.

**Tag removal:** In order to obtain a vector consisting only of valid content words, all tokens from $T_i$ that are part of a XML tag are removed. As all XML tags specified in a given WSDL document are predefined, the process of removing XML tags from the tokenized vector is simple.

**Word stemming:** With the help of Porter stemmer algorithm [22], only the relevant words in $T_i$ are reduced to their base words. Tokens among a common stem will generally have the similar meaning, for example, ‘establish’, ‘established’, ‘establish’, ‘establishing’, and ‘establishment’ all have the same stem ‘establish’. With respect to word deviations in the semantic of a Web service, using one or all of the tokens will not make a distinction. But, the words that appear frequently are more important when compared to others. The number of occurrences will be considered in the following steps.

**Function word removal:** Function words are said to be autonomous with respect to one another (Function words are the words we use to make our sentences grammatically correct. Pronouns, determiners, and prepositions, and auxiliary verbs are examples of function words). With the help of Poisson distribution to model word occurrence in documents [13], function words can be differentiated from content words. Using this step all function words from the service word vector can be removed. By calculating the overestimation factor for all words in the word vector, we can decide which word is a function word.

**Content word recognition:** Certain general computing content words such as ‘data’, ‘web’, ‘port’, etc. are typically present in WSDL documents. We cannot discriminate between Web services based on these words as they appear in most WSDL files. The goal of this step is to eliminate words which do not

---

*Figure 2. List of content words extracted from one web services after text-preprocessing step and stemming*
correspond to the specific semantics of the Web service. The k-means clustering algorithm (Jain, & Dubes, 1988) with \( k = 2 \) on \( T_i \) is then applied to cluster the remaining words into two groups such that one group corresponds to the meaning of the Web service whereas the other group is meant for general computing words.

Fig. 2 shows example of content words result from applying the above steps on WSDL of web service “medicalclinic_diagnosticprocesstimemeasure_service”. As many developers of web services do not follow camel case in naming the service or the variables that constitute the features inside the service, so we faced problems when making word splitting based on this rule. i.e the words in diagnosticprocesstimemeasur cannot be split. Also, some services have too many words describing it, so has long vector. see the following wordlist from service file medicalclinic_diagnosticprocesstimemeasure_service.wsdl.

3.2.2 Feature 2: Referenced Ontology
In SAWSDL version (Semantically Annotated WSDL) files, some semantic annotation has been added to refer to some well-known ontologies specially that define complex types as shown in Fig. 3. As we will see in our experiment how using this feature will highly enhance the accuracy of the clustering process.

```xml
<xs:complexType name="Book-Reference">
  <xs:sequence>
    <xs:element name="has-ISBN-number" type="xs:string"/>
    <xs:element name="has-publishing-house" type="Publishing-House"/>
    <xs:element name="has-place-of-publication" type="location"/>
    <xs:element name="has-title" type="xs:string"/>
    <xs:element name="has-author" type="generic-agent"/>
    <xs:element name="has-date" type="calendar-date"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Publishing-House">
  <xs:complexContent>
    <xs:restriction base="xs:string">
      <xs:simpleType>
        <xs:annotation>
          <xs:documentation>
            Reference of a publishing house.
          </xs:documentation>
        </xs:simpleType>
      </xs:simpleType>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Fig.3. Referenced Ontology and complex types elements in WSDL files

3.2.3 Feature 3: WSDL Types
WSDL documents contain a section that defines data containers which will be used by messages to transmit information between Web services. WSDL specifications use XML Schema Definition (XSD) as their canonical type system. Types can be as simple as a single element or as complex as an array of elements. Each element has a name attribute and a type attribute. While the name attribute is sometimes not a useful feature, the type attribute is a good candidate for describing the functionality of a service. The work [7, August] shows that complex data types are the most informative element in WSDL documents. We create the types feature for the Web services by extracting all the defined complexTypes along with their elements in each WSDL document.

3.2.4 Feature 4: Service Name
The composite name such as “cameraPriceMyShopService” can be split up into multiple names based on the assumption that a capital letter indicates the start of a new word. In our data set the words concatenated by underscore i.e. “camera_price_my_shop_service”.

4. EXPERIMENTS AND RESULTS

4.1 Experiment Design
In one experiment, we explored the resemblance among web services using WSDL document features such as WSDL content, complex types, referenced ontologies, and Web services name as appears in Fig. 1. Fig.4 shows the steps of the classification process that include text-preprocessing step for the data set to extract features, then narrowing these features and creating the features vector space model that will be used for the clustering step. Firstly, we compute the similarity of services with the help of semantic and TF-IDF information, which can be extracted from WSDL or RESTful. Secondly, we classified services by their similarities. The contents of the WSDL documents are parsed to produce the content word vector \( T_i \). The next step is to obtain a vector consisting only of valid content words without the XML tags. Our experiment is implemented using the tool RapidMiner1 and WVectorTool2 as a plugin in RapidMiner software to create vectors from the attached WSDL files. Feature extraction and integration is implemented using RapidMiner tool. The Term Frequency is calculated using the WV Tool.

---

1https://rapidminer.com/
2http://word-vector-tool.soft112.com/
4.2 Test Data Set

Our experiment is about supervised clustering of web services. The attached data set contains WSDL files that represent web services. The directory “services by domain” contains the services categorized by its domain, while the directory “services” contains all the services without categorization. We need to check after making clustering how much the classification process match the pre-defined categorization using two measures: precision and recall.

Our data set uses WSDL Service Retrieval Test Collection version 4.0 which consists of 1088 indexed WSDL services from the following 9 domains. The data set is divided to ratio of 25% to 75% between training data set and testing data set respectively, where the training data set was excluded from the test data set. The number of services and in each domain and their weights in the data set are as follows in Table 2. Table 2 shows the manual classification of the data set of the web services. By comparing the results of our experiment to this table, we can get measures about how much our clustering approach conform to the original classification, where the detailed measures will be introduced in the next section. A hundred percent clustering approach shall produce a result same as in table 2.

<table>
<thead>
<tr>
<th>Domain Name</th>
<th>Domain Size</th>
<th>Domain weight</th>
<th>Training Set Size</th>
<th>Test Set Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food</td>
<td>34</td>
<td>3%</td>
<td>7</td>
<td>27</td>
</tr>
<tr>
<td>Communication</td>
<td>58</td>
<td>5%</td>
<td>12</td>
<td>46</td>
</tr>
<tr>
<td>Education</td>
<td>285</td>
<td>26%</td>
<td>55</td>
<td>230</td>
</tr>
</tbody>
</table>

4.3 Classification Techniques

Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world. So rather than hand-coding software routines with a specific set of instructions to accomplish a particular task, the machine is “trained” using large amounts of data and algorithms that give it the ability to learn how to perform the task [10].

Three different clustering techniques used for aim of comparison according to accuracy and efficiency to know which technique is better. These techniques are Decision Tree, Naïve Bays, and Deep Learning. Also for the aim of getting which feature of WSDL file is more effective in getting better accuracy, a clustering is performed by Naïve Bays using WSDL content feature only in one experiment and using content, and referenced ontology feature in another experiment. In all cases, the accuracy of the result is measured using precision and recall values is recorded and compared.

4.3.1 Decision Tree Classifier

A decision tree is a classification model whose structure consists of a number of nodes and arcs. Many decision tree induction algorithms exist, the most popular being C4.5 and its variants [8]. The confusion matrix, a clean and unambiguous way to present the prediction results of a classifier, in Fig. 5 shows that the accuracy of this model is (85.63) with weighted average precision (81.27), weighted average recall (72.81) and F1-score (76.81). In Fig. 5, the column FP represent number of false positive (incorrectly predicted) services to measure precision. While, the row TP+FN represents correctly predicted and missed services to measure class recall, and the same concept is applied in next tables.

3http://semwebcentral.org/projects/owls2wsdl/
4.3.2 Naïve Bays Classifier

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. All naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable [9]. The confusion matrix, in Fig. 6 shows that the accuracy of this model is (90.34) with weighted average precision (87.35), weighted average recall (90.26) and F1-score (88.78).

4.3.3 Deep Learning Classifier

Deep learning is another Machine Learning (ML) algorithm. Deep learning is essentially a set of techniques that help you to parameterize deep neural network structures, neural networks with many, many layers and parameters. The confusion matrix, in Fig.7 shows that the accuracy of this model is (90.80) with weighted average precision (91.37), weighted average recall (91.11).

5. RESULTS Analysis & Comparisons

To evaluate the effectiveness of the algorithm, the experiment involved the computation of precision, recall, accuracy, and F1-score. Precision as in Eq.1 is the number of positive predictions divided by the total number of positive class values predicted. It is also called the Positive Predictive Value (PPV). Recall as in Eq.2 is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate. Precision measures the correctness of a classifier and recall measures the completeness of a classifier. High precision means that the algorithm has returned more relevant results. High recall means that the algorithm returned most of the relevant results.

\[
\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \quad \text{Eq. 1}
\]

\[
\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}
\]
Accuracy as in Eq. 3 measures the percentage of correct predictions done by the proposed model in comparison with actual measurements performed on test data. Accuracy is a weighted arithmetic mean of Precision and Inverse Precision (weighted by Bias) as well as a weighted arithmetic mean of Recall and Inverse Recall (weighted by Prevalence).

\[
\text{Accuracy} = \frac{\text{True positives} + \text{true negatives}}{\text{True positives} + \text{false positives} + \text{true negatives} + \text{false negatives}}
\]  

Eq. 3

The F1 Score is It is also called the F Score or the F Measure is a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score as in Eq. 4: Put another way, the F1 score conveys the balance between the precision and the recall.

\[
\text{F1-score} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})}
\]

Eq. 4

5.1 Classifiers Comparison

A detailed comparison between the three classifier results according to precision and recall at the level of class is shown in table 3. It is obvious that Deep Learning classifier has the best values for precision, recall, f1-score, and accuracy.

<table>
<thead>
<tr>
<th>Classifier Model</th>
<th>Decision Trees</th>
<th>Naïve Bays</th>
<th>Deep Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>Prsn. %</td>
<td>Ral. %</td>
<td>Prsn. %</td>
</tr>
<tr>
<td>Food</td>
<td>3.13%</td>
<td>100.00%</td>
<td>25.93%</td>
</tr>
<tr>
<td>Communication</td>
<td>5.33%</td>
<td>63.77%</td>
<td>95.65%</td>
</tr>
<tr>
<td>Education</td>
<td>26.19%</td>
<td>83.54%</td>
<td>88.26%</td>
</tr>
<tr>
<td>Medical</td>
<td>6.71%</td>
<td>61.67%</td>
<td>62.71%</td>
</tr>
<tr>
<td>Geography</td>
<td>5.51%</td>
<td>100.00%</td>
<td>80.00%</td>
</tr>
<tr>
<td>Economy</td>
<td>32.90%</td>
<td>95.05%</td>
<td>92.76%</td>
</tr>
<tr>
<td>Weapon</td>
<td>3.68%</td>
<td>100.00%</td>
<td>82.14%</td>
</tr>
<tr>
<td>Simulation</td>
<td>1.47%</td>
<td>40.00%</td>
<td>36.36%</td>
</tr>
<tr>
<td>Travel</td>
<td>15.07%</td>
<td>87.41%</td>
<td>91.47%</td>
</tr>
<tr>
<td>Weighted Prsn. / Ral.</td>
<td>86.78%</td>
<td>86.78%</td>
<td>90.5%</td>
</tr>
</tbody>
</table>

Accuracy: 85.63 % | 90.34 % | 90.80 %
F1-score: 86.8 % | 90.4 % | 91.8 %

Looking at Fig.8, we noted that Deep learning has the most stable precision across different classes weights (number of services relative to total data set).

![Classifiers Precision for Different Classes Weights](image)

The same applied in Fig.9, where Naïve Bayes classifier has the most stable recall values across different classes weights. From the two Fig.s, it is appearing that decision Tree Classifier recorded the most fluctuating scores for precision and recall according to class size.
Fig. 9. Classifiers recall across different classes.

Fig. 10 show weighted average precision and recall results between the three classifiers and it is obvious that Deep learning has the highest scores followed by Naïve Bayes with little differences.

Fig. 11. Naïve Bays classification accuracy using WSDL content feature.

Comparing Fig. 11 to Fig. 12, we see that the accuracy given by the same classifier by adding the new feature proposed by this research, referenced ontology, is (88.73%) with average precision (85.54%) and average recall (91.93%). This comparison indicates that the feature referenced ontology is key player in the accuracy of any selected classifier.

Table 4, record the time consumed in each model including, construction the model time, applying the model time, and total processing time. it was noted that deep learning techniques consume processing time more than naïve Bayes and decision tree techniques, so it is recommended to use Graphics Processing Unit (GPU) hardware with Deep Learning instead of CPU.

<table>
<thead>
<tr>
<th></th>
<th>Construct</th>
<th>Apply Model</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision</td>
<td>1,859.00</td>
<td>203.00</td>
<td>22,388.00</td>
</tr>
<tr>
<td>Naïve Bays</td>
<td>711.00</td>
<td>624.00</td>
<td>15,531.00</td>
</tr>
<tr>
<td>Deep</td>
<td>28,314.00</td>
<td>399.00</td>
<td>65,592.00</td>
</tr>
</tbody>
</table>

5.2 Features Comparison

As we compared various classifier methods to see which one has better accuracy, it is so important to know how much each WSDL feature contribute to the accuracy of the classification. Fig. 11 explains Naïve Bays classification accuracy details using one WSDL content feature only. The content feature represent vector for all distinct terms in the WSDL file and it gives accuracy (69.31 %) which is not so good.

Table 4

<table>
<thead>
<tr>
<th>Feature</th>
<th>Precision</th>
<th>Recall</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>prot.col</td>
<td>86.8</td>
<td>90.5</td>
<td>90.4</td>
<td>92.7</td>
</tr>
<tr>
<td>prot.type</td>
<td>85.5</td>
<td>85.5</td>
<td>80.0</td>
<td>80.0</td>
</tr>
</tbody>
</table>

Table 4. Processing Time Comparison
Table 5 and Fig.13 show detailed comparisons for various performance measures between three WSDL features used in this research: content, complex types, and referenced ontologies along with combination of them. The performance measures include precision, recall, and Kappa measure.

**TABLE 5**

**PERFORMANCE MEASURES FOR EACH WSDL FEATURE USING NAÏVE BAYS CLASSIFIER**

<table>
<thead>
<tr>
<th></th>
<th>Complex Types</th>
<th>Referenced Ontologies</th>
<th>WSDL Content</th>
<th>Combined Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted Mean Recall (%)</td>
<td>23.28</td>
<td>88.89</td>
<td>86.71</td>
<td>90.45</td>
</tr>
<tr>
<td>Weighted Mean Precision (%)</td>
<td>32.74</td>
<td>89.88</td>
<td>87.83</td>
<td>90.72</td>
</tr>
<tr>
<td>Classification Error (%)</td>
<td>76.9</td>
<td>11.15</td>
<td>13.45</td>
<td>9.66</td>
</tr>
<tr>
<td>kappa</td>
<td>0.17</td>
<td>0.86</td>
<td>0.83</td>
<td>0.88</td>
</tr>
</tbody>
</table>

The Kappa statistic (or value) is a metric that compares an Observed Accuracy with an Expected Accuracy (random chance). The kappa statistic is used not only to evaluate a single classifier, but also to evaluate classifiers amongst themselves. Kappa is means for evaluating the prediction performance of classifiers and it means that the classifier is in total agreement with a random classifier. Kappa always less than or equal to 1. A value of 1 implies perfect agreement and values less than 1 imply less than perfect agreement. In rare situations, Kappa can be negative. This is a sign that the two observers agreed less than would be expected just by chance.

Table 6 and Fig.14 compare the contribution of each WSDL feature used in this research into the classification process. Three dimensions of comparisons are used; (1) number of attributes the feature gives in the vector space, (2) the processing time required to process this features, and (3) the accuracy of classification result using this feature.

**TABLE 6**

**WSDL FEATURES CONTRIBUTION IN CLASSIFICATION PROCESS**

<table>
<thead>
<tr>
<th></th>
<th>Complex Types</th>
<th>Referenced Ontologies</th>
<th>WSDL Content</th>
<th>Combined Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Attributes</td>
<td>72</td>
<td>573</td>
<td>1168</td>
<td>1647</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>23%</td>
<td>89%</td>
<td>86%</td>
<td>90.3%</td>
</tr>
<tr>
<td>Processing Time (MS)</td>
<td>1663</td>
<td>792</td>
<td>741</td>
<td>2050</td>
</tr>
</tbody>
</table>

From table 6, it appears that using all features gives best accuracy but with more processing time, but using the new feature referenced ontology, gives less processing time with
acceptable slight decrease in accuracy. From the graph, it appears that the most balanced feature which compromise between number of attributes, processing time, and accuracy is the referenced ontology feature. Comparing to content and complex types features, it gives low number of attributes and low processing time with higher accuracy.

Fig.14. Radar plot for WSDL features contribution to performance measures

5.3 Comparison to Other Work
For the evaluation of our approach, we compared the classification results precision, recall, and accuracy with many other existing techniques. Table 7 shows this comparison according to weighted precision and recall. The weighted precision and recall multiplies the precision of each class into its weight relative to data set size, and the same applied to recall. Looking at the performance of our approach and existing web service clustering and classification approaches, we have noted that our approach yields high precision and recall above all methods except the approach of [14] and [19] which has higher precision and recall. But our approach works on larger data set 1088 web services distributed over 9 domains compared to 600 web services distributed for 8 domains in [19] and 69 services distributed over 5 domains in [14].

Table 7

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of services</td>
<td>400  (5 domains)</td>
<td>69  (5 domains)</td>
<td>116  (4 domains)</td>
<td>116  (4 domains)</td>
<td>600  (8 domains)</td>
<td>1088  (9 domains)</td>
</tr>
<tr>
<td>Weighted Precision</td>
<td>66.4 %</td>
<td>87.2 %</td>
<td>86.9 %</td>
<td>89.5 %</td>
<td>94.0 %</td>
<td>92.7 %</td>
</tr>
<tr>
<td>Weighted Recall</td>
<td>89 %</td>
<td>95.54 %</td>
<td>85.9 %</td>
<td>88.7 %</td>
<td>91.7 %</td>
<td>90.9 %</td>
</tr>
</tbody>
</table>

In addition, we have better accuracy using Deep learning as shown in table 8. For the processing time, we cannot compare as it is not reported by other work, but we claim that we have very competitive processing time using Naïve Bays classifier as shown in table 3.

Table 8

<table>
<thead>
<tr>
<th>Approach</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang, &amp; Pan, 2008 [20]</td>
<td>40 %</td>
</tr>
<tr>
<td>Konduri, 2008 [21]</td>
<td>70 %</td>
</tr>
</tbody>
</table>
6. CONCLUSION & CONTRIBUTION

This research presents a method for accelerating automated service discovery in web searches. We introduced service clustering for grouping services with similar functionality into classes according WSDL features. The aim of clustering is that the user query is compared with clusters by using matching algorithm (similarity measuring). Then the clusters are selected that have highest similarity with user query and user request compared only with whole web services available in the final selected clusters. This approach provides better performance in experimental results in terms of time complexity as we made enhanced text-preprocessing step before the extraction of the selected feature.

In this research, we used three normal WSDL features to process web services similarities, namely name, content, complex types, and introduced new feature named referenced ontology. Comparison between three features according accuracy and processing time are made and it is proved that the new introduced feature referenced ontology gives higher accuracy with lower processing time compared to the other features. Also, we applied Deep Learning technique as a supervised classification mechanism with ratio 20% to 80% between learning data set and testing data set respectively. The experiment compared the result to two other machine learning techniques, Naïve Bayes and Decision Tree. The Deep Learning technique gives higher accuracy but with more processing time compared to Naïve Bayes that has slight little accuracy.

The contributions of our work can be summarized in the following:

1. Introducing an approach for efficiently finding Web services on the Web by classifying we services into distinct clusters.
2. Collecting, analyzing and running several experiments on a large dataset of real world Web services.
3. Using Deep Learning as classification technique and proves that it gives higher accuracy comparing to Naïve Bayes and decision Tree.
4. Do more enhanced text-preprocessing before extracting the features of classification and that leads to more accurate results.
5. Introduce the new WSDL feature referenced ontology as dimension for web service classification and prove that it gives higher accuracy with low processing time compared to other features.

REFERENCES


**AbdelMoniem Helmy** born in Egypt 1977, graduated with B.Sc from Faculty of Computer & Information Science, Cairo University in 2000 and obtained Master degree (M.Sc) of Information Systems from the same faculty in 2005. He also obtained Diploma in e-Commerce from ITI in 2003 and another Diploma in Software Engineering from Fujitsu in 2006. Currently, AbdelMoniem is candidate for PhD at Institute of Statistical Studies & Research, Cairo University. He worked for many software projects and his research interests include software engineering, data analysis, semantic web, and web services.