Automatic Semantic Annotation of Web Documents

Milos Vujicic

A Thesis

In the Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Quality Systems Engineering) at
Concordia University
Montreal, Quebec, Canada

May 2009

Your file Votre référence
ISBN: 978-0-494-67266-2
Our file Notre référence
ISBN: 978-0-494-67266-2

ABSTRACT

Automatic Semantic Annotations for Web Documents

Milos Vujicic

Ontologies are the most important construct of the Semantic Web. From the first attempt of using simplified RDF syntax to the advanced features of the OWL languages, ontologies have arisen as the most viable technology offering solutions to integrate various Web resources into a more intelligent Web. The work presented in this thesis is a contribution to the new generation of the Web, which should be readable and interpreted not only by humans but also by machines, such as software agents. In order to allow ontologies to achieve their role of "animating" the traditional Web into this next generation Web, it is essential to find an efficient way to map all existent Web resources onto their corresponding ontology classes. In this thesis, we propose an approach for automatic semantic annotation of Web documents which is an effective way to make the Semantic Web a reality. Such an integrated Web would greatly improve the accuracy of search engines, bring a new generation of intelligent Web services, push the limits of multi-agent technologies and improve many other areas of human activity that we cannot even imagine today. Considering the size and the speed of the growing Web, it is clear that this task cannot be achieved manually. Semi-automatic and automatic annotations of Web documents using statistical text classification methods seem to be the most promising solution. This work is focused on an approach based on Naive Bayes text classification adapted to some characteristics that are particular to Web documents. A complete software solution is developed to allow testing feasibility of such an approach.

Furthermore, different variations of the text classification algorithms are tested and analysed in order to identify the most optimal approach to semantically annotate Web documents. Notably, the usage of Web documents hierarchy is explored as an option to improve the accuracy of semi-automatic and automatic annotations of Web documents. The results of each tested method are presented and commented. Finally, some aspects that could possibly be improved or approached in a different way are identified for future work.

Table of Contents

List of Figures

List of Tables

List of Equations

Chapter 1

Introduction

1.1 Context and Motivations

Since its inception, the Internet has grown incredibly fast. Such an extraordinary development has brought incredible opportunities in many areas, but also many concerns regarding its basic architecture. This chapter gives a brief overview of the current status of the Web and its limitations followed by an introduction to the most popular approaches to overcome these limitations. The last part describes the state of current efforts to improve the current Web and various issues related to these efforts. More specifically, the problem related to Semantic Web Annotation is introduced, which is the subject of this thesis.

The content of Internet as it is known today is entirely intended for human reading purposes and is purely display oriented. It is constructed without any notions of taxonomy and the categorisation of its content. In other words, Web browsers, Web servers and even search engines cannot distinguish a personal homepage from a major corporate Web site. The only way to classify and organize the Web is to perform keyword matching of its content. The disadvantage of this approach is the fact that it is impossible to find the Web pages related to a given subject of the search without finding many unrelated pages. For example, Google displays dozen of millions of pages related to the word "SOAP", which is a World Wide Web Consortium (W3C) standard for Web services. An important number of these pages are actually related to completely different subjects such as Soap Opera. To eliminate unrelated findings, it is always possible to add more search keywords. In our example, we could perform a search using Simple

Object Access Protocol keywords. That would definitely improve the results, but also eliminate all pages that are related to our subject without explicitly using the same keywords.

Another limit of the traditional Web is related to integration. Since there are no formal rules in the categorisation of the Web content, it is extremely hard to combine and aggregate its resources. For example, booking an airline ticket online can be a very tedious task. The first step is to find a few travel agencies selling tickets to our destination. Then we have to get the flight schedules and choose the most appropriate ones. Finally, we can choose the best price and book the ticket. In a perfect world, we should be able to accomplish the same task just by specifying our preferences regarding the dates, prices, destination and the rest would be performed automatically.

The Internet can be viewed as a huge distributed database. It is an important source of information and it has become a very important learning tool. Web data mining refers to the activity of getting useful information from the Internet. Since the current Web is made for human reading only, it is very difficult to build intelligent agents that can browse the Internet content in order to automatically synthesize and interpret various data. For example, we could have a Web site presenting the cities of California and another presenting the states of America. In today's Internet, only humans can deduce that the cities of California are actually the cities of America at the same time. This is a significant limit of the current Web and it is caused by the fact that it is not made to be read and automatically understood by computers.

1.2 Semantic Web Basics

The Semantic Web is the idea of having data on the Web defined and linked in a way that it can be used by machines, not only for display purposes, but also for automation, integration, and the reuse of data across various applications. The missing piece in the traditional Web that could allow the implementation of the Semantic Web is a metadata layer. In general, metadata is defined as "data about data"; it is data that describes information resources. To ensure that metadata can be automatically processed by machines, some metadata standards are needed. A standard is a set of agreed upon criteria for describing data. For instance, a standard may specify that each metadata record should consist of a number of predefined elements representing some specific attributes of a Web document, and each element can have one or more elements. This kind of standard is called metadata schema.

```
<html>
   <head>
      ...[document title here]...
      <meta name="DC.Date" content="(scheme=ISO8601) 2007-07-16">
      <meta name="DC.Title" content="Dublin Core Sample">
      <meta name="DC.Creator" content="Surname, Name">
      <meta name="DC.Creator.Address" content="e-mail@server.com">
      <meta name="DC.Subject.keyword" content="Dublin Core, Metadata">
      <meta name="DC.Type" content="User Guide, Tutorial">
      <meta name="DC.Identifier" content="(scheme=url) http://url/Name.html">
      <meta name="DC.Language" content="(scheme=ISO.639-1) sv">
   </head>
   <body>
      ...[document body begins]...
   </body>
</html>
```

Figure 1.1 Dublin Core Metadata

An example of a standard is the Dublin Core (DC) (Figure 1.1). It was developed in 1995 and it has 15 elements, which are called the Dublin Core Metadata Element Set. It is proposed as the minimum number of metadata elements required to facilitate discovery of document objects in a networked environment such as the Internet. Although the DC schema is a very simple example, it shows the key idea of adding metadata to a given document. The DC schema adds only general information to the Web documents and cannot be used to express complex semantic relations within various Web documents. In order to add semantic metadata to the Web, much more powerful tools are needed such as the RDF (Resource Description Framework) language.

1.3 RDF – Resource Description Framework

The RDF is an XML based language for describing information contained in a Web resource. A Web resource can be a Web page, an entire Web site, or any item on the Web that contains information in some form. RDF is considered to be the basic building block of the Semantic Web. The basic element of an RDF document is an RDF statement. It is used to describe the properties of Web resources and has the following format:

Resource (subject) + Property (predicate) + Property Value (object)

The property value can be a string literal or a resource. Therefore, in general an RDF statement indicates that a resource (the subject) is linked to another resource (the object) via a property (predicate). It can be interpreted as follows:

<subject> has a property <predicate>, whose value is <object>

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="http://SampleURL.net/Vehicle#">
   <rdf:Description rdf:about="http://SampleURL.net/ToyotaCorolla2008.rdf#ToyotaCorolla2008">
      <rdf:type rdf:resource="http://SampleURL.net/Vehicle#Sedan"/>
      <numberOfDoors rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</numberOfDoors>
  </rdf:Description>
</rdf:RDF>
```

Figure 1.2 RDF Statement

Figure 1.2 shows an example of an RDF statement that demonstrates how the rules explained above are implemented using the RDF syntax. The first line says that the document is in XML format. The second line indicates that the document is an RDF document. It shows the RDF namespace URI reference (http://www.w3.org/1999/02/22-rdf-syntax-ns#) and 'rdf' is used as a shortcut to represent this namespace. There is another namespace in the second line (http://SampleURL.net/Car#), which is the default namespace of the resource described in our RDF document. Therefore, any name that does not have a prefix in this document is assumed to be in this namespace. In the third line the 'rdf:Description' begins section where the resource is defined. In the same line there is a keyword 'rdf:about' used to identify the resource being described, which is 'http://SampleURL.net/ToyotaCorolla2008.rdf#ToyotaCorolla2008'. The fourth line, with the keywords 'rdf:type' and 'rdf:resource', specifies that the described resource is an instance of class 'Sedan', defined in the 'http://SampleURL.net/Vehicle#' namespace. The fourth line specifies that the 'Sedan' class has a property, named 'numberOfDoors', and for the resource 'ToyotaCorolla2008', the value of this property is '4'. In other words, the RDF document above should be read as follows: The resource 'http://SampleURL.net/ToyotaCorolla2008.rdf#ToyotaCorolla2008' has a property 'http://SampleURL.net/Vehicle#numberOfDoors', whose value is '4'.

When creating new resources, it is important to reuse those that already exist. This rule is applicable not only to the subjects and objects, but also to the predicates. In the previous example, instead of reinventing the resource Sedan, it was reused from the resource 'http://SampleURL.net/Vehicle#Sedan'. This implies that the resource 'http://SampleURL.net/ToyotaCorolla2008.rdf#ToyotaCorolla2008' represents exactly the same concept as 'http://SampleURL.net/Vehicle#Sedan'. Everything that has been added to this new resource is considered to be additional knowledge about it.

1.4 RDFS – Resource Description Framework Schema

The RDF Example shown in Figure 1.2 is a valid and compete RDF document that creates a new Web resource. But, it is obvious it cannot work without the class 'Sedan', which is declared somewhere else. The resource 'ToyotaCorolla2008' is an instance of class 'Sedan' and from the previous example, it is impossible to know what this class looks like. Moreover, it is impossible to know if there are super-classes or sub-classes of that class. It is impossible to know, if other properties can be defined, beside the property 'numberOfDoors'. Of course, it is acceptable not to reuse the existing resources and to reinvent the new vocabulary in each RDF document, but in that case none of these documents would be connected to the external world, which is exactly the principal objective of the RDF.

The RDFS is a language that one can use to create a vocabulary for describing classes, subclasses and properties of RDF resources. It is also used to associate the properties with the classes it defines. RDFS also defines the meaning of a given term by specifying its properties and what kinds of objects can be the values of these properties. In the case of the example with the resource 'Sedan', the vocabulary used for its definition could look like it is shown on Figure 1.3. This diagram shows that resource 'Sedan', used in the RDF example, is a sub-resource of another resource called 'Car'. It has a property called 'NumberOfDoors' which takes the value of the type 'http://www.w3.org/2001/XMLSchema#integer'. Besides 'Sedan', there are three other sub-resources of 'Car', which are: 'Coupe', '4X4' and 'Van'. At the same time, 'Car' itself is sub-resource of 'Vehicle'. 'Vehicle' has a property called 'OwnedBy' which takes a resource 'Driver'. The resource 'Driver' is a sub-resource of 'Person'.

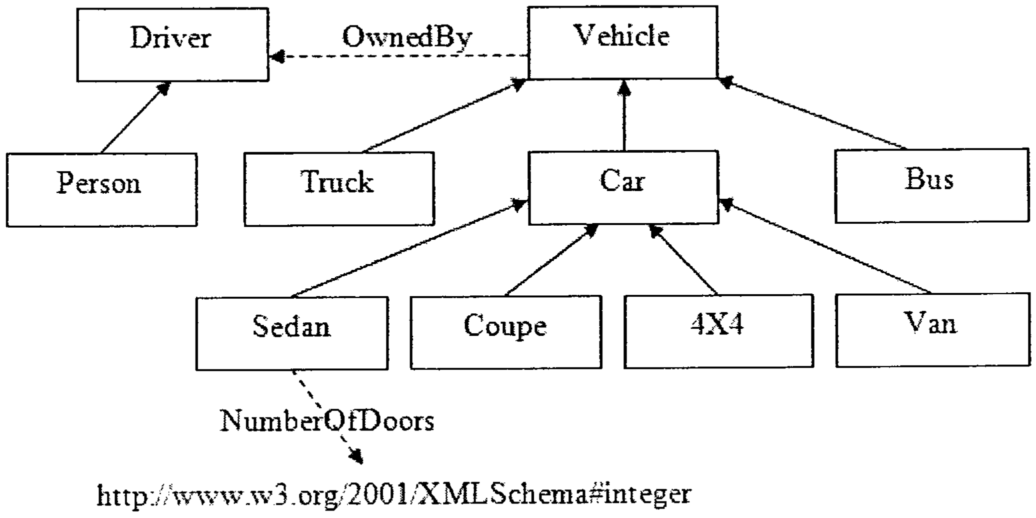


Figure 1.3 Ontology Vocabulary

The vocabulary presented in the diagram above, expressed using the RDFS syntax, looks like that shown on Figure 1.4. The RDFS syntax is relatively simple; the resources are defined using the keyword 'rdfs:Class' and sub-resources are identified with the keyword 'rdfs:subClassOf'. Properties are presented with the keyword 'rdf:property', whereas a property's parent resource is identified with the keyword 'rdfs:domain'. The keyword 'rdfs:range' defines the type of value that can be used for a given property. Figure 1.4 shows an example of RDFS document whose role is to define the vocabulary for creation of RDF resources, such as 'ToyotaCorolla2008'. This kind of vocabulary is known as *ontology* and represents one of the fundamental categories of the Semantic Web.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://SampleURL.net/Vehicle#">
   <!--Classes Definitions-->
   <rdfs:Class rdf:ID="Vehicle">
   </rdfs:Class>
  <rdfs:Class rdf:ID="Person">
  </rdfs:Class>
   <rdfs:Class rdf:ID="Truck">
      <rdfs:subClassOf rdf:resource="#Vehicle"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="Car">
      <rdfs:subClassOf rdf:resource="#Vehicle"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="Bus">
      <rdfs:subClassOf rdf:resource="#Vehicle"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="Sedan">
      <rdfs:subClassOf rdf:resource="#Car"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="Coupe">
      <rdfs:subClassOf rdf:resource="#Car"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="FourByFour">
      <rdfs:subClassOf rdf:resource="#Car"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="Van">
      <rdfs:subClassOf rdf:resource="#Car"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="Driver">
      <rdfs:subClassOf rdf:resource="#Person"/>
   </rdfs:Class>
   <!--Properties Definitions-->
   <rdf:Property rdf:ID="NumberOfDoors">
      <rdfs:domain rdf:resource="#Sedan"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
   </rdf:Property>
   <rdf:Property rdf:ID="OwnedBy">
      <rdfs:domain rdf:resource="#Vehicle"/>
      <rdfs:range rdf:resource="#Driver"/>
   </rdf:Property>
</rdf:RDF>
```

Figure 1.4 RDFS Ontology

1.5 OWL – Web Ontology Language

Although RDFS allows expressing quite complex relationships between various Web resources, it also has several limitations. For example, with RDFS it is impossible to indicate that two classes express the same concept. Even though the number one rule in RDFS is to reuse the existing resources, it is impossible to avoid a situation where someone creates a resource which already exists. In such a context, ability to recognize two or more Web resources as identical is extremely important. Another limitation of RDFS is related to the fact that it cannot express cardinality constraints. For example, in the case of 'ToyotaCorolla2008' resource, it would be possible to add more than one property 'numberOfDoors', which is totally meaningless.

For the above reasons and because of several other limitations, it was necessary to extend RDFS to allow the expression of more complex relationships between classes and to identify constraints on specific classes and properties. Thus, a new language was born with more expressive capability. The name of this language is OWL (Web Ontology Language) and it is the latest recommendation of W3C. It is probably the most popular language for creating ontologies today and it is built on RDFS. Therefore, all the classes and properties provided by RDFS can be used when creating an OWL document. The document below (Figure 1.5) shows the previous RDFS example expressed with OWL. Some new details have been added to this new ontology in order to show the extended expressive capacity of this tool.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://SampleURL.net/Vehicle#">
   <!--Classes Definitions-->
   <owl:Class rdf:ID="Vehicle">
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#OwnedBy"/>
            <owl:cardinality
               rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
               1
            </owl:cardinality>
         </owl:Restriction>
      </rdfs:subClassOf>
   </owl:Class>
   <owl:Class rdf:ID="Person">
   </owl:Class>
   <owl:Class rdf:ID="Truck">
      <rdfs:subClassOf rdf:resource="#Vehicle"/>
   </owl:Class>
   <owl:Class rdf:ID="Car">
      <rdfs:subClassOf rdf:resource="#Vehicle"/>
      <owl:equivalentClass
         rdf:resource="http://AnotherSampleURL.net/Automobile#Automobile"/>
   </owl:Class>
   <owl:Class rdf:ID="Bus">
      <rdfs:subClassOf rdf:resource="#Vehicle"/>
   </owl:Class>
   <owl:Class rdf:ID="Sedan">
      <rdfs:subClassOf rdf:resource="#Car"/>
   </owl:Class>
   <owl:Class rdf:ID="Coupe">
      <rdfs:subClassOf rdf:resource="#Car"/>
   </owl:Class>
   <owl:Class rdf:ID="FourByFour">
      <rdfs:subClassOf rdf:resource="#Car"/>
   </owl:Class>
   <owl:Class rdf:ID="Van">
      <rdfs:subClassOf rdf:resource="#Car"/>
   </owl:Class>
   <owl:Class rdf:ID="Driver">
      <rdfs:subClassOf rdf:resource="#Person"/>
   </owl:Class>
   <!--Properties Definitions-->
   <owl:DatatypeProperty rdf:ID="NumberOfDoors">
      <rdfs:domain rdf:resource="#Sedan"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
   </owl:DatatypeProperty>
   <owl:ObjectProperty rdf:ID="OwnedBy">
      <rdfs:domain rdf:resource="#Vehicle"/>
      <rdfs:range rdf:resource="#Driver"/>
   </owl:ObjectProperty>
</rdf:RDF>
```

Figure 1.5 OWL Ontology

The OWL syntax is very similar to the syntax of RDFS. Thus, classes and subclasses are declared with the same keywords. Properties are also declared in similar way, except for the fact that in OWL there is a distinction between properties connecting a resource to another resource and properties connecting a resource to an 'rdfs:Literal' or an XML schema built-in data type value. Properties connecting to a resource are declared with the keyword 'owl:ObjectProperty' whereas the properties pointing to an 'rdfs:Literal' or an XML schema built-in data type value are identified with 'owl:DatatypeProperty'.

The most important difference between RDFS and OWL is OWL's capacity to express more complex relationships between Web resources. Figure 1.5 shows an example of a cardinality constraint within the 'Vehicle' class. This is done by using 'owl:Restriction', 'owl:onProperty' and 'owl:cardinality' keywords. Thus, a 'Vehicle' can be owned by only one 'Driver'. Another example of OWL's additional capability is shown in the 'Car' class. The keyword 'owl:equivalentClass', indicates that the class 'Car' is equivalent to the class 'Automobil' located at http://AnotherSampleURL.net/Automobile. Besides these two examples, there are many others OWL keywords that can be used to express much more detailed and complex relationships between resources.

1.6 Semantic Web Services

Before discussing how Web services could benefit from the Semantic Web, it is important to understand current Web service architecture and its standards. A Web service is an application (server) that provides a Web-accessible API, so that another application (client) can invoke it programmatically. The standard communication protocol used for accessing Web services is Hypertext Transfer Protocol (HTTP). The obvious reason for this standard is the fact that HTTP is everywhere. Any machine that can run a Web browser supports HTTP. Firewalls normally allow HTTP traffic; so it is possible to use HTTP to talk to any machine. Web services are a relatively new solution in software integration and adopted standards are still evolving. However, WSDL, SOAP and UDDI seem to be well adopted standards by all major players in Web services industry. WSDL (Web Services Description Language) is an XML-based language used for describing Web services. A WSDL document is automatically generated by any Web service, describing its all public methods as well as all the input and output data types used by these methods. SOAP is a protocol implemented over the HTTP protocol used to transfer data between Web services. This protocol ensures data transmission between the applications running on different servers, regardless of the programming languages and platforms that are used. Since SOAP is layered over HTTP, it inherits all its advantages. UDDI acts as a Web services registry. Its main function is to provide support for finding and publishing services description. It has a directory structure where businesses can register and search for Web services.

Web services have brought numerous advantages in software integration. The ability to integrate applications running on different platforms and developed with different programming languages is definitely the most important advantage of using Web services. The first step in a software integration process, implicating Web services, is finding the appropriate Web service. UDDI is the ultimate tool made for Web service discovery. At present, UDDI classification schemas facilitate the discovery of Web services, but its categorizations still do not allow an unambiguous Web service definition. Thus, even when the appropriate Web service seems to have been found, there is no guarantee that it is actualy true. This is because two Web services can have exactly the same methods and be related to the same subjects, but give completely different types of outputs. Therefore, before its integration, some additional tests and technical analysis must be done. The semantic Web gives us the tools to possibly solve this problem. The principal idea is to add semantics to Web services so they could be unambiguously defined. This could be achieved by using ontologies, which have already been introduced in this document. Solving the problem of ambiguous categorization and taxonomy of Web services could greatly improve their discovery process. At the same time, it would bring many other opportunities in Web services integration. If it is possible to find the appropriate Web service with no mistake, why not make this process automatic. The next step is to invoke the Web service automatically. Ontologies can be used to describe Web services but, they can also be used to describe in detail a Web service's operations, which would allow their invocation on the fly. Introducing the Semantic Web in the world of Web services, gives the possibility of going even further. Automatic composition of existing Web services to obtain a new functionality is another objective that could be reached. Quite often, a specific business need requires several Web services to work together. If it is possible to discover and invoke them automatically, their composition could be done automatically as well. At present, the three most popular Semantic Web services approaches to reach the objectives explained above are:

- Web Ontology Language – Services (OWL-S)
- Web Services Modeling Ontology (WSMO)
- Web Service Description Language Semantics (WSDL-S)

1. OWL-S Upper Ontology to Describe Web Services

OWL-S is written using OWL language. It could be defined as an upper ontology used to semantically mark-up Web services. An upper ontology is not related to any particular knowledge domain, its role is to provide some common and general information about a given Web service. OWL-S is made of three sub-ontologies whose role is to describe the following aspects of Web services:

- Profile ontology describes what service does. This ontology is mainly used to advertise the service, thereby enabling a service requester to determine whether the given service meets the needs or not.
- Process ontology describes how service works. More precisely, it describes the procedures necessary to interact with the service from the client's point of view.
- Grounding ontology describes how the service is invoked. This ontology provides terms that can be used to describe how the service can be accessed technically.

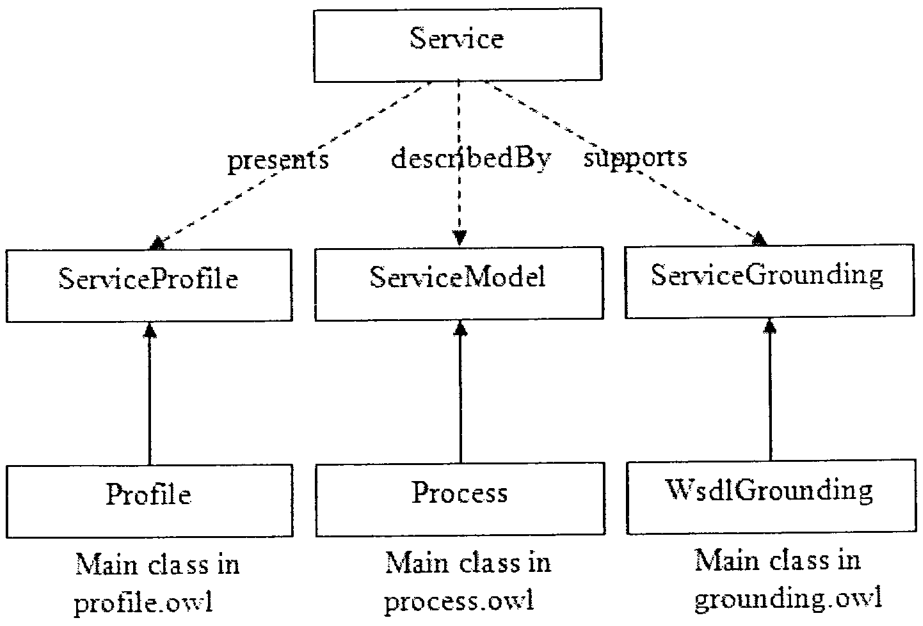


Figure 1.6 OWL-S Service Ontology

The three OWLS-S sub-ontologies are connected together through a final higher-level ontology, called Service ontology (Figure 1.6). The role of this ontology is to describe the semantic relationships among the other three ontologies.

2. WSDL-S Semantic Annotations

Besides the OWL-S approach which is based on creating stand-alone semantic descriptions of Web services based on OWL ontology, there are also other approaches that focus on reusing and extending existing structures. One of these approaches is WSDL-S approach introduced by IBM and the University of Georgia. This method adds a semantic layer to the WSDL document, which has been accepted by W3C as a standard since 2001. WSDL-S depends on domain-specific ontologies. Its semantic annotations are added to different parts of a WSDL document by using domain ontologies. Another advantage of WSDL-S is that it does not limit the choice of the language in which the domain specific ontology is constructed. This approach mainly focuses on dynamic discovery of Web services. However, it does not provide enough semantic information for automatic invocation and composition of Web services. To solve the problems of automatic discovery, the service description must be published in some registry such as UDDI. The Organisation for the Advancement of Structured Information Standards (OASIS) has published a recommended mapping schema that implements the mapping of WSDL-S onto UDDI data structure.

3. WSMO Web Service Modeling Ontology

WSMO provides a complete framework allowing semantic description of Web services. It is an ongoing research and development initiative which consists of three following domains of activity:

- WSMO, which provides formal specification of concepts for Semantic Web services.
- WSML (Web Services Modeling Language), which defines the language for WSMO concepts.
- WSMX (Web Services Execution Environment), which defines and provides reference implementation allowing the execution of Semantic Web services.

The WSMO approach can be summarized in four top level concepts: Ontologies, Web services, Goals and Mediators. WSML is used as the modeling tool in each of these four concepts. As in the two previous approaches, the fundamentals of the WSMO are Ontologies. The language used for building ontologies in the WSMO is WSML.

1.7 Current State and Future Improvements

It is commonly recognized that the Semantic Web would solve many issues and bring new possibilities to numerous computer related fields. At present, there are still many Semantic Web aspects that have to be improved. However, like any other computer technology, the Semantic Web will never stop evolving and will never reach the point where everything would be perfect.

1. RDF related issues

At present, almost all Semantic Web concepts are subject to criticisms and even the basic foundations are not an exception. For instance, a RDF statement, the basic building block of OWL ontology language, has been criticized due to various technical aspects. Here is a list of some points that are subject to criticisms related to RDF [12]:

- It is impossible to distinguish an RDF node element from a property element by simple inspection of the element in question.
- The frame-style approach of the description block does not clearly match the RDF model - triples in the RDF graph.
- There are excessive choices for users in choosing how to write RDF/XML.
- Elements, attributes and attribute values are used for the same purposes, for example, encoding an RDF URI reference.
- The way that XML QNames are used does not constrain the elements and attribute tags that can appear in RDF/XML.
- The unconstrained syntax cannot be described completely with XML schema languages such as DTDs and W3C XML Schema (WXS).
- It does not allow using xsi:type for specifying W3C XML Schema data types.
- The syntax is not easy to use with XML technologies such as XSLT, XQuery and other XML tools (mostly due to the unconstrained tags and many abbreviations).
- It is impossible to embed RDF/XML in XHTML while retaining DTD validation (while this is also true for any other XML syntax embedded in a DTD-constrained format).
- It is hard to emit human-readable RDF/XML from an RDF graph due to the range of choices
- RDF/XML cannot describe collections of literals.
- Not all property URIs can be encoded.

As it might be observed from the above criticisms, the issues reported to RDF are mainly of technical nature. Although they indicate some important problems, they do not seem to be unsolvable.

2. Ontologies development and evolution

Although there already exist large-scale ontologies, ontology engineers are still needed to construct the ontology and knowledge base for a particular task or domain, and to maintain and update the ontology to keep it relevant and up-to-date. Manually constructed ontologies are time-consuming, labour-intensive and error-prone. Moreover, a significant delay in the update of ontologies causes currency problems that actually hinder the development and application of the ontologies [13]. The majority of existing ontologies have been generated manually. Generating ontologies in this manner has been the normal approach undertaken by most ontology engineers. For this reason, researchers are looking for other alternatives to generating ontologies in a more efficient and effective way. The starting point for creating an ontology could arise from different situations. An ontology can be created from scratch, from existing ontologies (whether global or local ontologies) only, from a corpus of information sources only; or a combination of the latter two approaches. Various degrees of automation could be used to build ontologies, ranging from fully manual, semi-automatic, to fully automatic. At present, the fully automatic method only functions well for very lightweight ontologies in very limited circumstances. At present there are several approaches to automate ontologie creation. In [13], there are summarized descriptions of the major approaches. However, none of them have solved all potential problems. Advancement in this area would definitely help a more rapid introduction of Semantic Web.

3. Implementing semantic annotations

Considering the scale and dynamics of the worldwide Web, the largest knowledge base ever built, it becomes clear that it is impossible to annotate Web documents manually. Thus, introducing automatic or semi-automatic methods for semantic annotations of Web resources comes into view as a logical and obvious solution to this problem. At the same time, knowing that similar methods have been used by the most popular Web crawlers for categorisation purposes, it is questionable if these approaches can perform a better job in the context of Semantic Web. There are a few elements that have to be considered to justify using automatic and semi-automatic approaches for Semantic Web annotations. First of all, it is true that problems related to the Web resources categorization in the current Web are due to the errors caused by the limits of automatic methods. However, the primary problem of the current Web lays in the categorization itself, which is based only on keywords. By introducing ontologies for classification purposes, these approaches would perform a much better job. Still, it can be difficult to accept the idea of using approaches that inevitably must generate defects. At the same time, manual annotation can not be considered safer. When performed manually, semantic annotations get highly dependent on personal preferences of individuals. Also, the level of knowledge related to semantic annotation of Web resources is not equal from one person to the other, which could create some important reliability related issues. For all these reasons, automatic and semi-automatic semantic annotations should not be considered just as a possible solution but rather as a necessity.

1.8 Contributions and Thesis Outline

The subject of this thesis is automatic and semi-automatic semantic annotation of Web documents. It is particularly focused on the usage of text classification algorithms to recognize ontology class instances among documents that belong to a Web domain. Although several aspects related to the classification of Web documents are introduced, particular attention is given to feature selection and the hierarchy of documents. An approach for feature selection is proposed using the Porter Stemming algorithm along with a new algorithm for the text classification exploiting the hierarchy of Web documents. In addition, a software application is developed for automatic annotation of Web documents. Results of various analyses, from the data gathered with this application are presented to describe advantages and disadvantages of the different approaches.

The thesis is divided into five chapters. After introducing the Semantic Web in Chapter 1, in Chapter 2, automatic and semi-automatic semantic annotations are presented in detail explaining the difficulties and limitations of that approach as well as the details of the method used in this thesis. Chapter 3 describes the functionalities of the software application developed for automatic annotation of Web documents. The results and analyses of the tests related to the different approaches are presented in Chapter 4 and at the end, a conclusion and some propositions for future work are presented in Chapter 5.

Chapter 2

Automatic Semantic Annotation of Web Resources

2.1 Introduction

This chapter introduces the principles of statistical document classification and its possible application in the Web context. In the next two sections, an introduction to the basics of the Web document classification is given along with some important issues that have to be resolved. The fourth part of this chapter describes in detail the approach used in this work as well as the data set that is used for tests and data analysis. The fifth section describes in detail the algorithms used for the statistical text classification and the last two sections focus on feature selection and the hierarchy of documents. As already mentioned, these two aspects of Web document classification are in fact the principal research area of this thesis.

2.2 Web Documents Classification

The Semantic Web is still in its early development stage and has a lot of unfinished features. As shown in the previous chapter, some fundamental aspects such as ontology languages, development and evolution of ontologies, as well as semantic annotations of documents are not an exception. In this work, the main research goal is automatic and semi-automatic semantic annotation of Web documents. Since, all the areas of semantic Web are highly interconnected, a significant breakthrough in any of them could bring completely new approaches in all other areas. However, this work is focussed solely on semantic annotations of Web documents and only the current standards from other areas will be considered.

Although, there are significant differences between automatic annotations for the traditional Web and the annotations for the Semantic Web, some of the basic principles and approaches are the same. In both cases, classification algorithms are needed to determine the type or category of a given Web document. That job is basically done through the evaluation of Web resource's text content by applying different mathematical and statistical methods.

In the case of the traditional Web, the objective of this classification process is to find and rank the relations between Web resources and various subjects or keywords. In the case of the Semantic Web, instead of subjects or keywords, we speak rather about ontologies, ontology classes or ontology attributes. Relations between these new categories and Web resources are determined within ontologies themselves.

As a starting point in this work, some existing resources have been considered which focus on similar area of researches ([59], [15], [51], [56], [31], [36], [44]). In all these works, the main objective is the semantic annotation of Web documents using automatic and semi-automatic methods. The statistical text classification is another point shared across all these approaches. In these and many other reference papers, various algorithms for text classification have been analysed providing valuable information about their characteristics. In an important number of these works, the Naive Bayes Classifier algorithm has been tested and proven to be a very good choice for document classification [22], [29], [23]. This algorithm is also used in this thesis and its implementation is more detailed in the second part of this chapter.

There are many factors that influence the efficiency of a text classification approach. The most important factors are the algorithms used for the text classification and many reference papers provide valuable results in that domain ([17], [18], [27]). Although various findings from these papers have been used in this thesis, the main area of research is focused on Feature Selection and the Hierarchy of Web Documents. These two aspects also have a great impact on the efficiency of a text classification approach. Some important reference papers related to these areas have also been used in this thesis ([18], [31]).

2.3 Problem Formulation

As already mentioned, automatic and semi-automatic methods for semantic annotations are not just a possible direction for a full implementation of the Semantic Web, but rather a necessity. First, converting the current traditional Web to the semantic one, with no automatic or semi-automatic methods, would require a manual updating of all current Web pages currently available on the Internet. In other words, each html document should be manually tagged as an instance of one or more ontology classes. Of course, this kind of global software conversions would not be the first in the history of computer science. The year 2000 bug would be a good example of the software adaptation to the new global conditions. However, there are some significant differences between the millennium bug software conversion and an eventual implementation of the Semantic Web. In the case of the millennium bug, non-correction of a piece of software was simply not an option, unless it was already at end of its life. On the other hand, introducing the Semantic Web will not eliminate the traditional Web. They can coexist together and at present it is almost impossible to imagine a reason that would motivate people all around the world to add the semantic annotations to their HTML documents.

Another important reason why automatic and semi-automatic approaches are required is related to the accuracy and consistency of the semantic annotations. They would be required, even if the Semantic Web was already reality where all HTML documents are semantically annotated by their owners. For multiple reasons, human performed annotations cannot be always trusted or taken as the only source of information. Even in the traditional Web, there are many examples where people try to cheat the Web crawlers by adding keywords unrelated to their Web pages, in order to improve their visibility. These kinds of issues will definitely exist in the Semantic Web. In addition to this, there are also human errors as well as omissions that have to be taken into account. For instance, HTML documents can be related to many different ontology classes. When manually annotated, some of these relationships might not be expressed.

In summary, regardless of which technologies will be used to bring the Semantic Web into reality, the usage of automatic and semi-automatic methods seems to be an inevitable choice for semantic annotation of HTML documents. Of course, manual annotations will largely facilitate this job, but they will definitely be insufficient to create a basis for a robust and consistent Semantic Web.

Automatic and semi-automatic annotations for the text content of the Web pages have already been the subject of several academic studies. In all of them, the text classifying algorithms have been used as the principal approach for identifying and classifying textual content of HTML documents. These algorithms have already been implemented in spam detecting software with a high degree of success. However, in the case of the Semantic Web annotations, the same success is still lacking.

Machine learning is the primary element of all text classification approaches [47]. The efficiency of a machine learning algorithm depends on the quality of data that are used for training purposes. For instance, in the case of spam detecting software, examples of unsolicited e-mails must be used in order to train the software for recognizing this kind of documents. Basically, the same technique is used in the case of Web documents classifying.

In the case of the spam detection, the learning process is controlled by end-users. As soon as an e-mail is identified as a spam, the same e-mail can be easily traced and eliminated in any inbox of the same e-mail server. Then, the detected unsolicited e-mail is added to the training set of data which will allow the software to eliminate these kinds of defects in the future automatically.

The process of classifying HTML documents for semantic annotating purposes is different and more complex in many regards. First of all, it is not just about detecting the defects. It is about assigning a document to the right category with the fewest possible defects. Here the term defect is referred to assigning a document to the wrong category. The number of categories is equal to the number of ontology classes describing the domain for which the HTML pages have to be classified.

Another important difference is in the way the training data is provided. In the case of Web resources classification, training data sets have to be gathered and classified manually. For each ontology class, a set of related HTML documents must be provided in order to teach the software how to recognize the same kinds of documents. Of course, recognizing a document category cannot be one hundred percent accurate and it is a serious limitation of this approach, because these defects have to be detected within an additional process. Although promising, all recent research has still not succeeded in significantly reducing the relative number of defects. Hence, the main objective of this work is to identify and understand the main causes of these defects and to try to find solutions or possible ways to improve the overall process of the semantic annotation of Web documents.

2.4 Overview

One of the most important aspects in any study is having results that can be compared with the results from other similar studies. In the case of document categorization and recognition, this can be very challenging given that the results of different approaches are literally incomparable unless the input data are the same. In this work, the data used in all tests has been chosen from a previous study [15], which is publically available at http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/. This data set contains HTML pages collected from computer science departments of various universities in January 1997 by the World Wide Knowledge Base (Web->Kb) project of the CMU text learning group. The 8,282 pages were manually classified into the following categories: Student (1641), Faculty (1124), Staff (137), Department (182), Course (930), Project (504) and Other (3764).

The class 'Other' is a collection of pages that were deemed not to be the "main page", For example, a particular faculty member may be represented by a home page, a publications list, résumé and several research interests' pages. Only the faculty member's home page was placed in the 'Faculty' class. The publications list, résumé and research interest's pages were all placed in the 'Other' category. For each class the data set contains pages from four universities: Cornell (867), Texas (827), Washington (1205), Wisconsin (1263). Remaining 4120 pages, labelled as Miscellaneous in the dataset, are collected from several other universities. The ontology diagram below (Figure 2.1) shows the relations between different classes which are part of the data set used in this work.

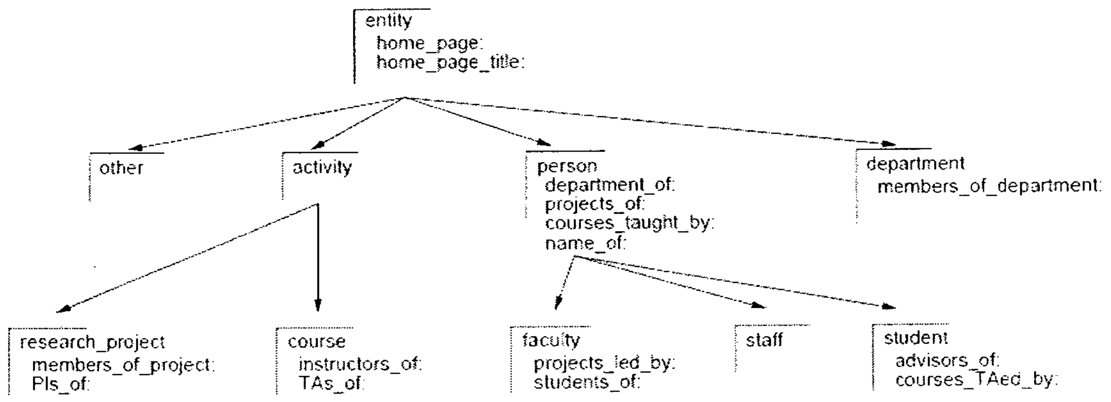


Figure 2.1 University Ontology

2.5 Statistical Text Classification

Recognizing the type of a document based on its textual content is basically a question of statistical calculations performed on words found in it. The statistical methods used for classifying pages involve using the so-called bag of words. The text appearing anywhere on a HTSML page is taken into account, but also the words that occur in the title and

HTML headings of the page as well as the words that occur in hyperlinks that point to other pages. The approach involves building a probabilistic model for each defined class using classified training data, and then classifying new pages by selecting the class that is most probable given the evidence of words describing the new page. The method used for classifying Web pages is based on the Naive Bayes classifier. The probabilistic model used in this classifier ignores the sequence in which the words occur. Such models are often called unigram or bag-of-words models because they are based on statistics about single words in isolation. Since the unigram model naively assumes that the presence of each word in a document is conditionally independent of all other words in the document, this approach, when used with Bayes Rule, is often called naive Bayes. The conditional independence assumption is clearly violated in real-world data, however, despite these violations; empirically the Naive Bayes classifier does a good job of classifying text documents [22], [29], [23].

In this thesis, Naive Bayes classifier is used with minor modifications based on Kullback-Leibler Divergence [15]. Given a document d to classify, a score is calculated for each class as follows:

$$Score_c(d) = \frac{log\,Pr(c)}{n} + \sum_{i=1}^{T} Pr(w_i \mid d) log\left(\frac{Pr(w_i \mid c)}{Pr(w_i \mid d)}\right)$$

Equation 2.1 Document Score

In this formula n is the number of words in document d, T is the number of words in the vocabulary (discussed later in this chapter) that exist in a given document, and w_i is the ith word defined for T. $Pr(w_i|c)$ represents the probability that a randomly drawn word from a randomly drawn document in class c will be the word w_i. $Pr(w_i|d)$ represents the proportion of words in document d that are word w_i. The class predicted by the method for a given document is simply the class with the greatest score. This method makes exactly the same classifications as Naive Bayes, but produces classification scores that are less extreme.

Estimating the word probabilities, $Pr(w_i|c)$ is the principal element in Naive Bayes. Basically, by calculating the product of word probabilities for a given type of document, it is possible to recognize, with a certain level of confidentiality, if a documents belongs to that type. One of the problems that have to be solved in this approach is due to assigning zero probability to words that do not occur in the training data for a particular class. For instance, even if many words strongly indicate that a given document is possibly an instance of a given ontology class, only one document word that is not a part of the training data would give a probability zero, which once multiplied with other probabilities would result in zero as a total probability. There are several smoothing methods that can be used to solve this problem. In this work, it has been achieved by calculating the overall probability for a given document using only the words that exist in both, the document and the training data. The probability calculated this way can be expressed as follows:

$$Pr(w_i \mid c) = \frac{N(w_i, c)}{T_c + \sum_j N(w_j, c)}$$

Equation 2.2 Word Probability

In this formula, $N(w_i, c)$ is the number of times word w_i occurs in the training data for class c and T_c is the total number of unique words in class c.

2.6 Feature Selection

In text recognition and classification algorithms, it is extremely important to decide what should be considered as a word and what words are worth considering. As a first approximation, a word could be defined as sequence of characters delimited by an empty space or an EOF (end of the line) character. However after a little more analysis, it is easy to realize that all comas, periods, semicolons, question marks, quotes are attached to the words which have to be removed as well. Of course, these are the easiest ones, but what about the dashes and slashes? How to distinguish a compound word from two words separated with a dash? How to consider the URLs and e-mail addresses or phone numbers? In this work, all these special characters are treated as word delimiters. Also, all the "'s" at the end of words are eliminated before being stored into the database. It is true that this approach eliminates some important parts of information (obviously a compound word does not have the same meaning as two separate words). However, the approach used in this work is based on Naive Bayes algorithms and one of its principal assumptions is that the presence (or absence) of a particular feature is unrelated to the presence (or absence) of any other feature. So by removing the special characters a text of course loses some important parts required for its understanding, but from the statistical point of view it is rather insignificant.

Besides the special characters, there are also words that are considered to be of no significant value for text recognition algorithms. They occur evenly in any text regardless of its subject. These words are called stop words and in this work the following words have been excluded in the case of all ontology classes: a, an, and, are, as, at, be, but, for, have, has, I, if, in, is, it, me, not, of, on, or, out, so, that, the, there, they, this, to, was, with, you, your. In addition, all words made of only one character have been removed as well.

In English as well as in all other European languages, a word can be inflected into many different words. For instance, the word connect can is used to make words such as connected, connecting, connection, connections. Even though the meaning of these words is not the same, they all refer to the same semantic concept. Therefore, reducing them to their root form can be advantageous for two reasons. First, the total number of words in a document would be reduced which improves the performance of text recognition algorithms. Secondly, differences in writing styles used in same kinds of documents is reduced which improves the success rate in the text recognition algorithms. The process for reducing inflected words to their root or stem forms is called stemming and the reduced form of words are known as stems. The stem does not need to be identical to the morphological root of the word. It is sufficient that related words map to the same stem, even if this stem is not in itself a valid root. At present, there are many stemming algorithms available for various purposes. The algorithm used in this work is known as the Porter Stemming algorithm [16]. This algorithm has been developed by Martin Porter and it is publicly available for reusing. It is widely used and became the de-facto standard algorithm used for English stemming.

Another very important topic in estimating document classes is related to the definition of the domain vocabulary. The objective at this stage is to find a set of words which are the most representative for delineating differences between the available ontology classes. Basically, the perfect word would be a word that is systematically present in one particular class of documents and systematically absent in all other classes. So it becomes easily obvious that some very common words for all classes are in fact the worst candidates. For instance, in the case of the University ontology, the words such as university, student, courses, department or faculty are not the best candidates. The most valuable words can be easily found mathematically from the training data set. For each word, a score representing its statistical value is calculated then the words with the highest scores are selected for the domain vocabulary. This is achieved using the following expression [15]:

$$I(C,W_i) = \sum_{v_i \in \{w_i, \neg w_i\}} \sum_{c \in C} Pr(c,v_i) \log\left(\frac{Pr(c,v_i)}{Pr(c)Pr(v_i)}\right)$$

Equation 2.3 Domain Vocabulary

In the formula above, v_i is a random variable indicating whether word w_i is present or absent in a document. Thus, $v_i \in \{w_i, \neg w_i\}$ for the values it takes in a document. The C is a random variable taking values of all the class labels, $c \in C$. $Pr(c)$ is the probability of the class in the training data set, which can be calculated by dividing the number of documents representing the class c with the total number of the documents. $Pr(v_i)$ represents the probability that a word is found in a document from the training data set, regardless the class it belongs to. It can be obtained by dividing the number of documents having a given word with the total number of the documents. Finally, $Pr(c, v_i)$ is the probability that a given word is found in a particular document class. It can be found by dividing the number of documents belonging to a particular class that contain a given word with the total number of documents belonging to the same class.

Once, these scores are calculated for all words, it is important to decide on the vocabulary size. In this work, tests have been performed with different vocabulary sizes in order to find the most appropriate value. Most of the studies done on text classifications suggest using smaller vocabulary sizes in order to achieve better results.

2.7 Hierarchy of Documents

The University Ontology shown on Figure 2.1, like majority of ontologies, has a tree structure. Two levels containing more than two ontology classes can be identified in the case of the University Ontology. The first level is made of 'Other', 'Activity', 'Person' and 'Department' ontology classes, where as at the second level, there are two separate groups of classes. The first group is made of 'Research Project' and 'Course' classes, which are the sub-classes of the 'Activity' class. The second group contains 'Faculty', 'Staff' and 'Student' classes which are the sub-classes of the 'Person' class. As a result, the initial ontology can be divided into 3 different subsets or domains and perform document classification independently for each domain. This approach divides documents into more homogenous groups which can be more efficient while classifying texts that have very similar words. From a technical point of view, the same mathematical calculations can be used as in the non-hierarchical approach. However, feature selection has to be performed for each domain separately as well as various probability calculations. In this approach, features selected in the lower domains can be significantly different from those selected in the first level. In the process of the text classification, a document is first evaluated against the domain in the first level. If the document is identified as an instance of the 'Department' or 'Other' class than the classification process ends at that point. On the other hand, if the document is identified as an 'Activity' or 'Person' instance then it has to be evaluated one more time against the domain containing the sub-classes of the class chosen at the first level.

Another approach consists of keeping the same hierarchical separation of the University ontology. The difference from the approach described in the previous paragraph is in the way a class of a given document is evaluated at the first level. Instead of having four ontology classes: 'Person', 'Activity', 'Department' and 'Other', the first two classes are actually replaced by their sub-classes. Thus, the number of classes at the first level is equal to the total number of University ontology classes. In this way, evaluating the class of a document at the first level is exactly the same as in the non-hierarchical approach. As a result, right after the first level class evaluation, the ontology classes from all domains are already estimated. Re-estimating documents being evaluated as the instances of the second level classes is a part of the next step of this approach. This step could be seen as process of refining. It is exactly the same as the process performed at the first hierarchical approach. This second approach is in fact a mix of hierarchical and non-hierarchical approach. Implementing and analysing both approaches gives more detailed information about efficiency of hierarchical versus non-hierarchical approach.

Chapter 3

Semantic Annotation Tool

3.1 Introduction

The objective of this chapter is to describe the software developed in this thesis, which is used for all the experimental tests and data analyses described in the following chapters. The next section of this chapter briefly introduces various technologies used to develop the tool. The following three sections give respectively the detailed information about three main modules of the tool: the software training module used to train the software to semantically annotate Web documents, the module responsible for annotation of new Web documents and the tool's settings module used to enable or disable various software settings. The last three sections are intended to describe the technical side of the tool. In the Data Model section, there is an overview of the entity-relationship model. The Performance Considerations section describes some performance considerations that have to be taken into account with this kind of tool and last section describes some of the most important program units that were used to implement the algorithms described in the previous chapter.

3.2 Technical Environment

The main part of this work is the Semantic Annotation Tool, developed for identifying and classifying HTML resources. This software has been developed for the research purposes of this work. It was developed on J2EE platform, using the following open source software:

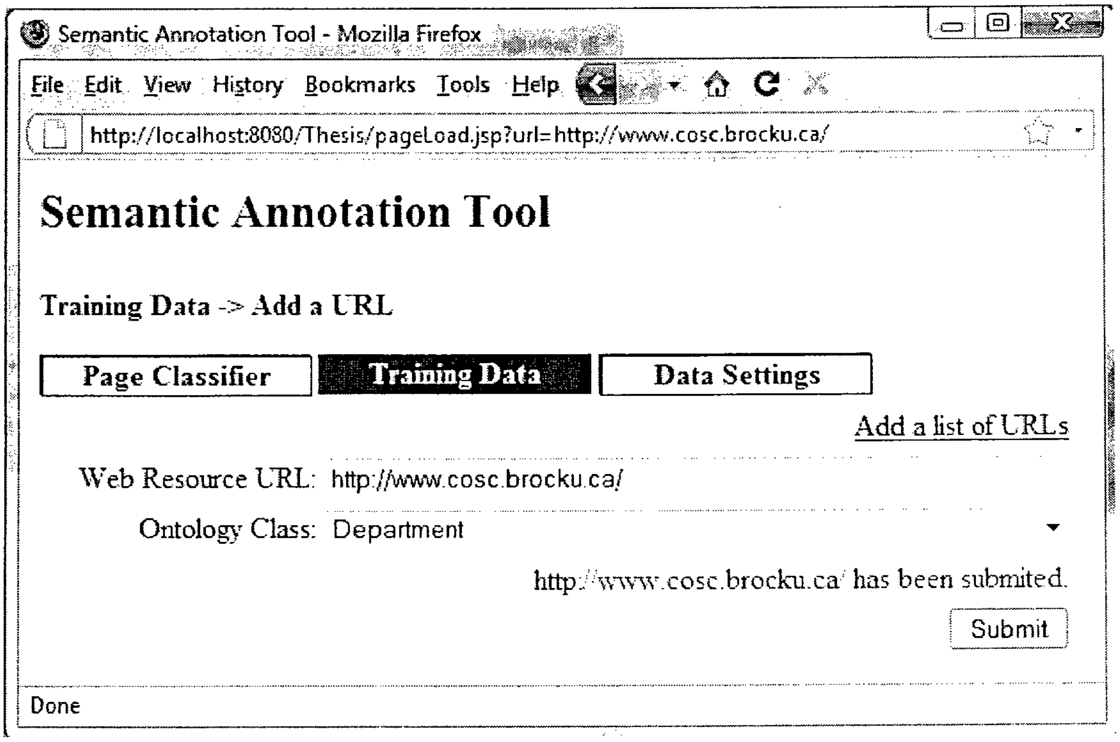
- MySql database for data storing purposes

- SQL language for program units that access stored data and implement all major statistical and mathematical algorithms
- HTML Parser, used for parsing HTML documents
- Java language used for developing the main software layer that integrates together SQL program units, HTML Parser and the presentation tier
- HTML and JavaScript for development of the presentation tier
- Apache Tomcat Web server

The Semantic Annotation Tool is made of three modules: the 'Page Classifier', the 'Training Data' and the 'Data Settings'. All the aspects and topics of the semantic classification of Web documents discussed in Chapter 2 are implemented in three modules described in the following sections.

3.3 Software Training

The role of the 'Training Data' module is to load pre-classified pages in order to teach the software how to recognize new pages belonging to the same ontology classes. The main screen of this module (Figure 3.1) provides the interface for uploading Web pages by providing their addresses at the 'Web Resource URL' text field and selecting the corresponding ontology class from the 'Ontology Class' list. Once the page is uploaded and classified, a message in red appears confirming successful completion. In the example below, the page http://www.cosc.brocku.ca/ (Brock Computer Science) has been uploaded as an instance of the department class. The limitation of this option is that only one page can be specified at a time. Therefore, the option 'Add a list of URLs' has been add to this module in order to allow the specification of a file containing the list of URLs that have to be classified. It is exactly the same as the previous option, except that instead of a URL, a file location has to be specified. In addition to selecting the ontology class from the list, in this option it can also be specified in the file for each URL. If both options are chosen, the one from the file has the priority. This option was used for loading 8282 HTML documents from the World Wide Knowledge Base project. The entire process of parsing and classifying these pages takes around one hour.


Semantic Annotation Tool - Mozilla Firefox

http://localhost:8080/Thesis/pageLoad.jsp?url=http://www.cosc.brocku.ca/

Semantic Annotation Tool

Training Data -> Add a URL

Web Resource URL: http://www.cosc.brocku.ca/

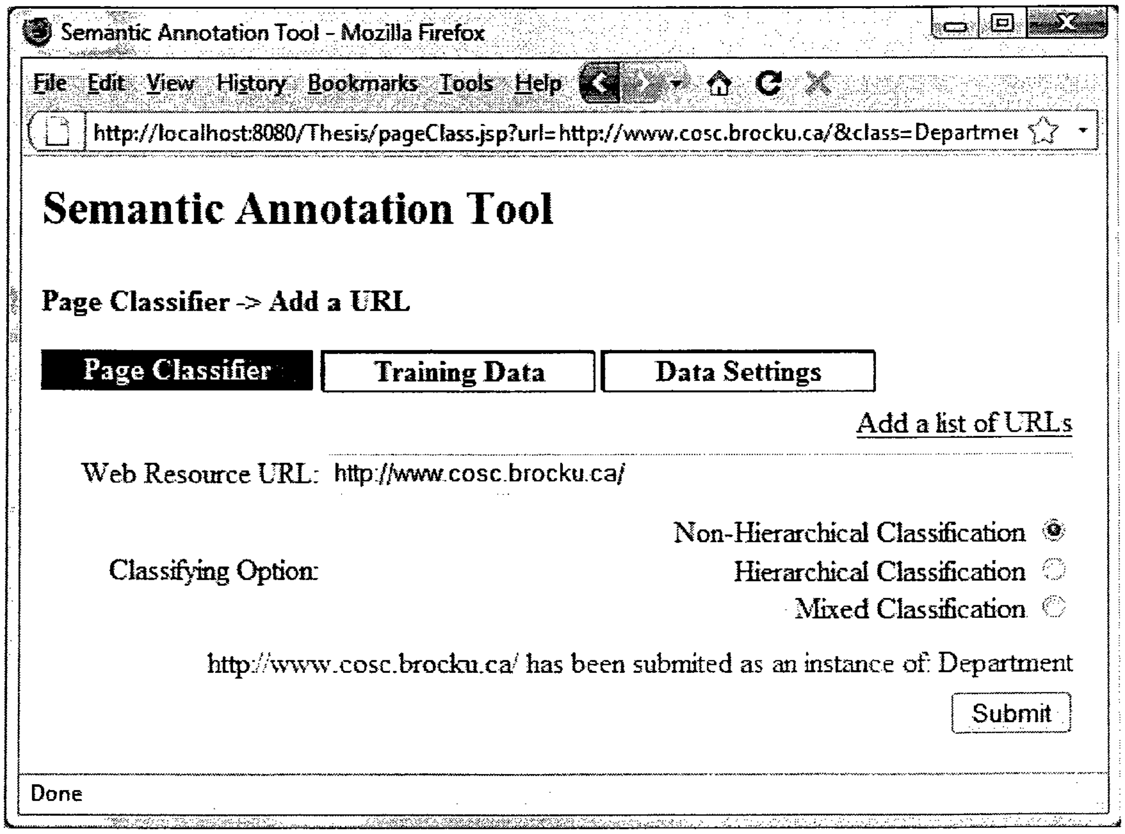
Ontology Class: Department

http://www.cosc.brocku.ca/ has been submited.

Figure 3.1 Semantic Annotation Tool – Training Data

3.4 Recognizing Ontology Class Instances

The 'Page Classifier' module is used for loading and identifying ontology classes of unclassified HTML documents (Figure 3.2). In this module as well, there are two options for uploading HTML documents. They work in exactly the same way as those described in the previous module. The only difference is that in this module there are no lists to specify ontology classes. When a new page is successfully uploaded and classified, a message in red characters appears giving the information about the ontology class that has been assigned by the software. In the example below, the page http://www.cosc.brocku.ca/ (Brock Computer Science) has been recognized by the software as an instance of the department class. The option 'Add a list of URLs', available in both modules described above, is very useful for loading large numbers of documents. It also allows integration of the Semantic Annotation Tool with Web crawling tools, since most of them can generate URL lists of found Web resources.


Semantic Annotation Tool - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Page Classifier
Training Data
Data Settings
Add a list of URLs
Submit
Done

http://localhost:8080/Thesis/pageClass.jsp?url=http://www.cosc.brocku.ca/&class=Departme

Semantic Annotation Tool

Page Classifier -> Add a URL

Web Resource URL: http://www.cosc.brocku.ca/

Classifying Option: Non-Hierarchical Classification [✓] Hierarchical Classification Mixed Classification

http://www.cosc.brocku.ca/ has been submited as an instance of: Department

Figure 3.2 Semantic Annotation Tool – Page Classifier

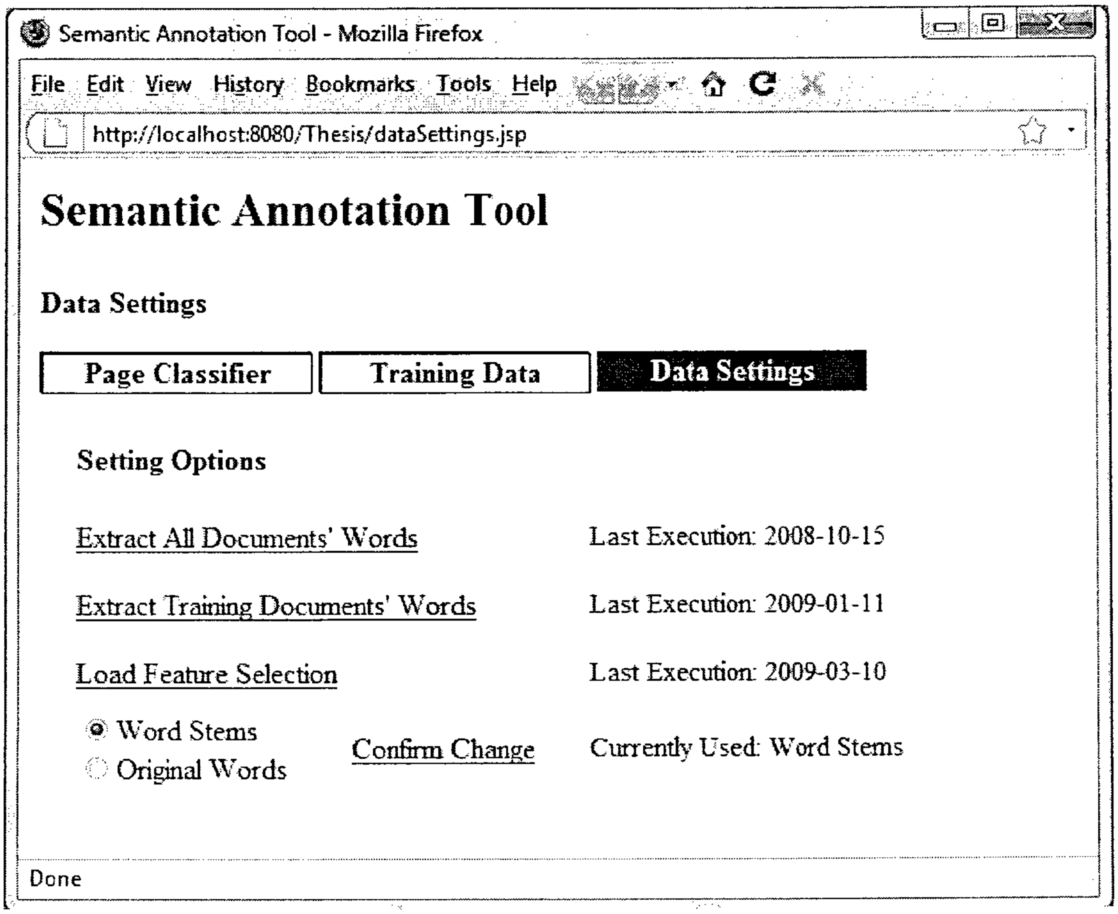
As discussed in this thesis, HTML documents can be classified using three different methods: Non-Hierarchical classification, Hierarchical classification and Mixed Classification. Any of these three classification approaches can be chosen by selecting the corresponding radio-button at the 'Page Classifier' module.

3.5 Software Settings

The third module 'Data Settings' contains the options that have to be configured once the training dataset is uploaded in order to prepare the software for recognizing new HTML documents (Figure 3.3). There are five options available in this module:

- The 'Extract All Documents' Words' option is used to extract the words from all the documents that are already uploaded and stored in the database. It is the only option that does not have to be executed after loading the training data, because it is performed implicitly while parsing HTML documents. This option has to be executed only if some other system parameters have been changed after uploading the training data. Examples of such parameters are the stop words or the stemming algorithm (introduced in Chapter 2).
- The 'Load Training Documents' Words' procedure identifies the words that belong to the documents used for the training purposes. During this process, all unique words are eliminated as their statistical value is with no significance. Since it uses the words extracted by 'Extract All Documents' Words', it must be executed after that procedure.
- The 'Load Feature Selection' is used to identify the words (features) for each domain vocabulary of a given ontology. This procedure creates one vocabulary per domain. It also calculates all the probabilities and statistics required for the document

classification algorithms. This option must only be executed after 'Load Training Documents' Words' procedure had been executed, since it uses the data generated by that procedure.


Semantic Annotation Tool - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Done

http://localhost:8080/Thesis/dataSettings.jsp

Semantic Annotation Tool

Data Settings

Page Classifier | Training Data | Data Settings

Setting Options

Extract All Documents' Words — Last Execution: 2008-10-15

Extract Training Documents' Words — Last Execution: 2009-01-11

Load Feature Selection — Last Execution: 2009-03-10

[✓ Word Stems] Original Words — Confirm Change — Currently Used: Word Stems

Figure 3.3 Semantic Annotation Tool – Data Settings

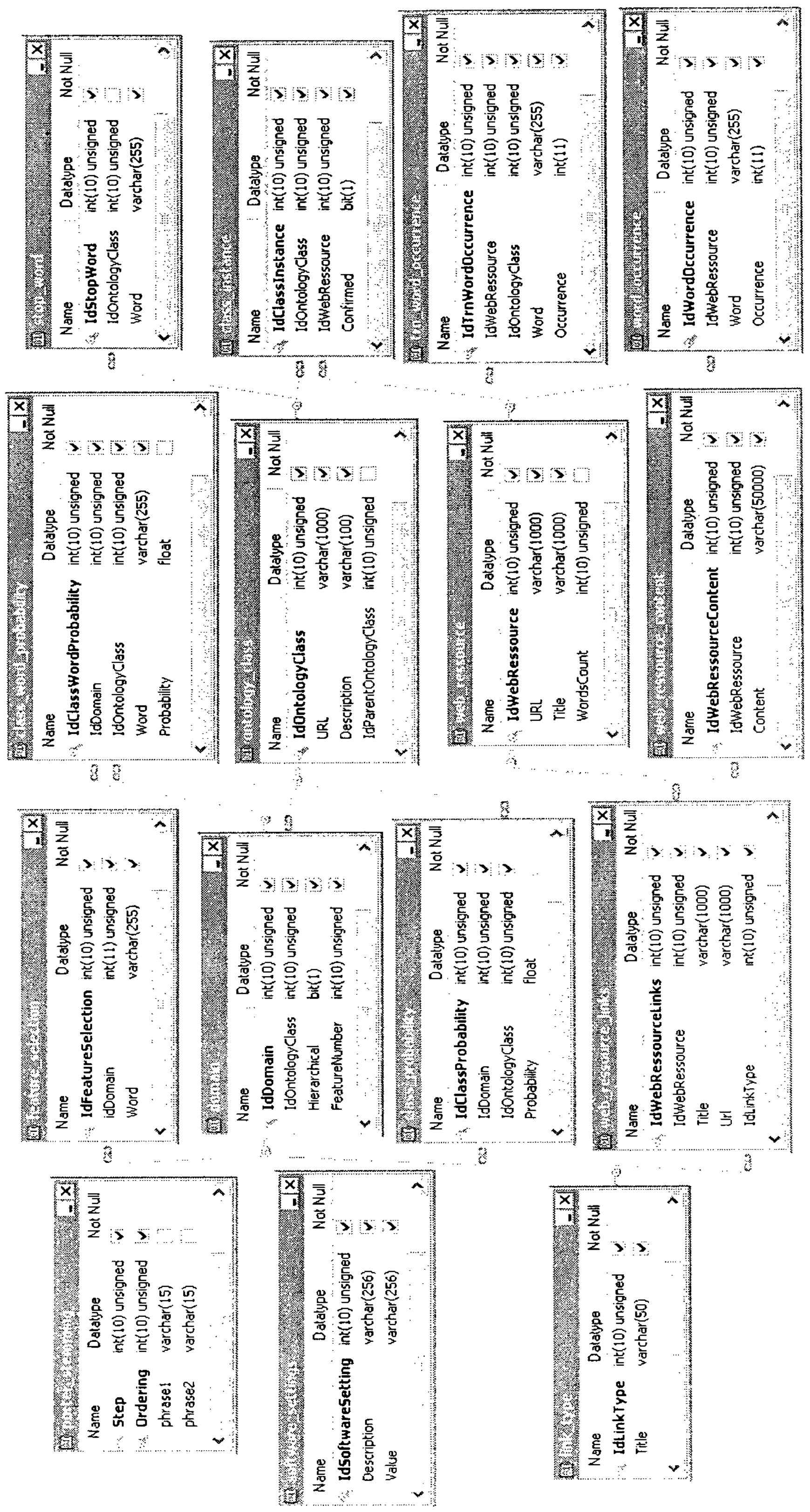
- The Semantic Annotation Tool can use original words or word stems. This choice should be made before loading the training data because switching from one option to the other requires the re-execution of the procedures previously described.

3.6 Data Model

The entity-relationship data model on Figure 3.4 shows the database structure used for storing all data required by the Semantic Annotation Tool. This data model is made of eight SQL tables:

- The 'web_ressource' table contains the basic information for all stored documents, such as URL, title and the total number of words.
- The table 'web_ressource_content' is used to store the whole textual content of Web documents. It is populated right after the successful parsing of a Web document.
- The 'word_occurrence' table is used to store words as well as the number of times they occur on each HTML document. It is populated from the data stored in 'web_ressource_content', which allows the software to reload the words from stored documents without parsing the HTML document again.
- The 'ontology_class' table contains basic information about of the ontology classes, such as URL and Description. During the training phase, the content of this table will determine for which ontology classes the software can be trained.
- The 'class_instance' is used to link a Web resource to the appropriate ontology class. The column 'Confirmed' in this table is used to differentiate between the training data and the documents discovered by the software. This information could have also been stored in the 'web_ressource' table, given the fact that a Web page can be indentified as instance of only one ontology class. However, in the real world, this is not true and for that reason, the table 'class_instance' is added to the data model.
- The table 'stop_word' contains the words that are considered to be of no significant value for the text recognition algorithm. Although, the relationship "zero to many" with the 'ontology_class' table allows storing different words for each class, in this work all the stop words are the same for all ontology classes.

- The table 'web_ressource_links' is used to store the links, as well as the text associated with them for each stored HTML documents. Same as with the 'web_ressource_content' table, the 'web_ressource_links' is also loaded during the page parsing phase.
- The table 'link_type' contains different types of links that can be found on the Web pages such us: http, https, ftp, e-mail, irc and javascript. Although they are a part of the model, the tables 'web_ressource_links' and 'link_type' are not really used by the text classifying algorithms.
- The domains or the ontology subsets used for hierarchical text classifications are stored in the 'domain' table. The number of features used by classifying algorithms for each domain is stored in this table as well.
- Various software variables, such as last execution times of each procedure or the parameter determining whether the software uses the stemmed words or not, are stored in the table 'software_settings'.
- The Porter stemming algorithm requires some data which has to be permanently stored. For that purpose the table 'porter_stemming' has been created. It contains some basic rules used for word stemming.



stop_word

Name	Datatype	Not Null
IdStopWord	int(10) unsigned	✓
IdOntologyClass	int(10) unsigned	☐
Word	varchar(255)	✓

class_instance

Name	Datatype	Not Null
IdClassInstance	int(10) unsigned	✓
IdOntologyClass	int(10) unsigned	✓
IdWebRessource	int(10) unsigned	✓
Confirmed	bit(1)	✓

trn_word_occurrence

Name	Datatype	Not Null
IdTrnWordOccurrence	int(10) unsigned	✓
IdWebRessource	int(10) unsigned	✓
IdOntologyClass	int(10) unsigned	✓
Word	varchar(255)	✓
Occurrence	int(11)	✓

word_occurrence

Name	Datatype	Not Null
IdWordOccurrence	int(10) unsigned	✓
IdWebRessource	int(10) unsigned	✓
Word	varchar(255)	✓
Occurrence	int(11)	✓

class_word_probability

Name	Datatype	Not Null
IdClassWordProbability	int(10) unsigned	✓
IdDomain	int(10) unsigned	✓
IdOntologyClass	int(10) unsigned	✓
Word	varchar(255)	✓
Probability	float	☐

ontology_class

Name	Datatype	Not Null
IdOntologyClass	int(10) unsigned	✓
URL	varchar(1000)	✓
Description	varchar(100)	✓
IdParentOntologyClass	int(10) unsigned	☐

web_ressource

Name	Datatype	Not Null
IdWebRessource	int(10) unsigned	✓
URL	varchar(1000)	✓
Title	varchar(1000)	✓
WordsCount	int(10) unsigned	☐

web_ressource_content

Name	Datatype	Not Null
IdWebRessourceContent	int(10) unsigned	✓
IdWebRessource	int(10) unsigned	✓
Content	varchar(50000)	✓

feature_selection

Name	Datatype	Not Null
IdFeatureSelection	int(10) unsigned	✓
IdDomain	int(11) unsigned	✓
Word	varchar(255)	✓

domain

Name	Datatype	Not Null
IdDomain	int(10) unsigned	✓
IdOntologyClass	int(10) unsigned	✓
Hierarchical	bit(1)	✓
FeatureNumber	int(10) unsigned	✓

class_probability

Name	Datatype	Not Null
IdClassProbability	int(10) unsigned	✓
IdDomain	int(10) unsigned	✓
IdOntologyClass	int(10) unsigned	✓
Probability	float	

web_ressource_links

Name	Datatype	Not Null
IdWebRessourceLinks	int(10) unsigned	✓
IdWebRessource	int(10) unsigned	✓
Title	varchar(1000)	✓
Url	varchar(1000)	✓
IdLinkType	int(10) unsigned	✓

[illegible]

Name	Datatype	Not Null
Step	int(10) unsigned	✓
Ordering	int(10) unsigned	✓
phrase1	varchar(15)	☐
phrase2	varchar(15)	☐

software_settings

Name	Datatype	Not Null
IdSoftwareSetting	int(10) unsigned	✓
Description	varchar(256)	✓
Value	varchar(256)	✓

link_type

Name	Datatype	Not Null
IdLinkType	int(10) unsigned	✓
Title	varchar(50)	✓

Figure 3.4 Data Model for Semantic Annotation Tool

Statistical text classification involves complex mathematical calculations and as such it can be very challenging in the case of several ontology classes with a lot of pages used for training purposes. In this work, more than 8000 pages have been used with almost two million words in total. In such a context, software performance becomes a very important aspect. Since the calculations are performed on data stored in a relational database, SQL language was the most appropriate choice to implement all major algorithms. Furthermore, some additional SQL tables have been added to the data model shown on Figure 3.4, in order to store some data that takes long time to calculate, but that does not change frequently. For instance, all calculations involving only the training data have to be performed only once. The following tables contain this type of data:

- The table 'class_probability' stores the information about probability or proportion of documents for each ontology class within the training data set.
- The table 'class_word_probability' contains probabilities or distributions of words across the ontology classes.
- The table 'feature_selection' is used to store the words chosen for each domain vocabulary.
- The words selected for training purposes are stored in 'trn_word_occurence'. All the words stored in this table can be found in the 'word_occurence' table, but the opposite is not true, because of the unique words which are not used for training purposes.

The information in these three tables could have been accessed through the SQL queries or views, since it is calculated upon previously stored data. However, having them physically stored significantly improves the performance of the software.

3.7 Program Units

Besides the SQL tables, there are various SQL functions and procedures that are used in order to fully implement the features of the Semantic Annotation Tool. The complete source code of these program units is available in the annex of this document. Here is a non-exhaustive description of the most important ones:

- The functions: 'evaluate_document_class', 'evaluate_document_class_hier' and 'evaluate_document_class_mix' are used to evaluate documents' ontology classes in the 'Page Classifier' module (Figure 3.2). They respectively implement Non-Hierarchical, Hierarchical and Mixed Classification. Despite the different algorithms used in these classifications, they are all based on the same statistical calculations described in Equation 2.1 (in Chapter 2).
- The procedure 'extract_web_ressource_words' is used to extract the words of a given Web page using the rules and limitations already described (see Feature Selection). This procedure depends on the implementation of several functions, for example the Porter stemming algorithm [16]. This procedure implements the features of the 'Training Data' module (Figure 3.1).
- Compared to the previous procedure, 'extract_all_web_ressource_words' is almost the same, except that it is used for re-loading the words of all the documents already stored instead of loading only one document at a time. This procedure is used for

implementation of the first option 'Extract All Documents' Words' of the 'Data Settings' module (Figure 3.3)

- The procedure 'extract_trn_word_occurrence' is used to reload words used as training data into a separate SQL table. The option 'Extract Training Documents' Words' in the 'Data Settings' module (Figure 3.3) is implemented using this procedure. This feature is basically used to improve the system performance and to add more flexibility when different training data sets have to be produced from the preloaded Web documents.
- Probably the most important procedure, from the point of view of statistical text categorization, is 'load_feature_selection'. It is used when executing the 'Load Feature Selection' option from 'Data Settings' module (Figure 3.3), which is in fact an implementation of the formulas shown in Equation 2.2 and Equation 2.3 (Chapter 2).

Chapter 4

Experimental Results

4.1 Introduction

This chapter details the results of the various tests and analysis performed in this work. The next section briefly describes the data set on which the tests were performed. The subsequent sections explain in detail the following topics: feature selection analysis, words vs. word stems classification, hierarchical method and mixed method. While feature selection analysis focuses more on the feature frequency or how often a given feature can be found in a randomly drawn document, remaining sections focus attention on the efficiency of different approaches in Web document classification.

4.2 Tests Description

Different types of tests and analysis have been performed using the data from the CMU text learning group. As suggested by the members of that group, the Semantic Anotation Tool was trained on three of the universities plus the misc collection, and tested on the pages from a fourth, held-out university. The universities chosen for the training purposes are Washington, Wisconsin and Texas, whereas Cornell University is taken for testing the software. This choice is not related to any particular characteristic of the dataset and is purely random. The issue with using the pages from only one university is the fact that there is only one Web page in such dataset that belongs to the 'Department' ontology class. In order to add more statistical significance to the test results related to that class, the Web pages from the computer science departments of 26 Canadian universities have been added to the testing data. The added pages have been found using the Google search tool by using the keywords "computer science department" and using the option “pages from Canada”. From the result set of approximately 30000 pages, the first 26 pages that were actually the main pages of a computer science department were added to the testing data set.

In all tests, the class 'Other' has obtained the worst results, which is normal because this class contains documents of various types. In an ideal world, these pages would be classified into additional ontology classes, which would be represented in the training data set. However, it is obvious that some Web documents will always be unclassified or be a part of the 'Other' class, since the development of ontologies is conditioned by the development of the Web content. Although very often it is possible to anticipate the evolution of a given ontology, it is impossible to create perfectly suitable categories for something that has not been created yet. Therefore, the presence of the class 'Other' in the University ontology is quite realistic. Nevertheless, in the case of the dataset used by CMU text learning group, the content of the class 'Other' also includes the pages that are in fact the instances of the real classes, but are classified as Other because they are not the main pages [15]. Consequently, the accuracy achieved in classifying documents of this class is very low.

4.3 Feature Selection Analysis

Once the stop words are eliminated and all the rules used for feature selection are applied (such as word stemming), there are around 48000 different words or features representing the dataset used by CMU text learning group. By eliminating the features that occur in only one document, the number of features is reduced to approximately 23000. Additional elimination of the least occurring features further reduces the number of the remaining features, but at a slower rate. For instance, by eliminating the features occurring in only two documents or less, the total number of remaining features becomes 16000. Figure 4.1 shows number of documents represented by different number of features which follows an exponential distribution. It can be seen that a small proportion of the features have high frequency, namely about 500 features occur in a little more than in 400 documents. In following section of this chapter, it can be seen that the most accurate results are obtained with the vocabulary sizes that are in the zone close to the mean of this distribution (715 features for the dataset made of stemmed words excluding the stop words).

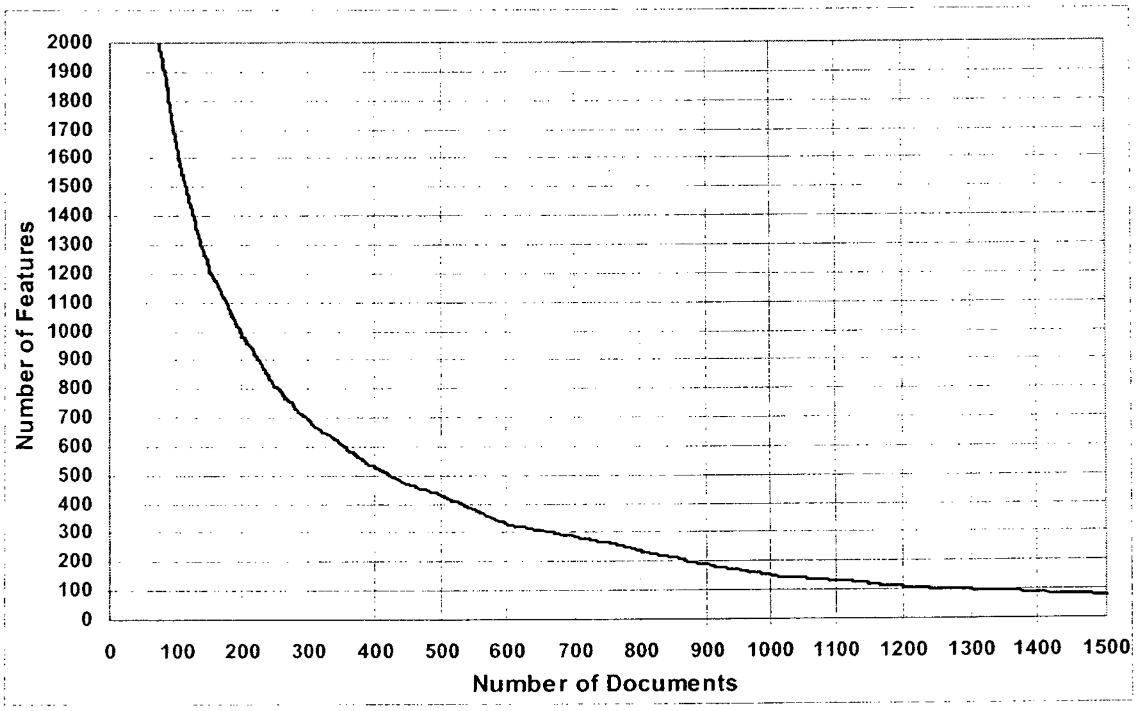


Figure 4.1 Frequency of the CMU Dataset Features

4.4 Words vs. Word Stems Classification

To compare properly the efficiency of these two text classification methods, several tests were performed with the same data set, using different vocabulary sizes. The performance of each of these tests was measured by calculating the percentage of pages assigned to a given class that are actually members of that class (accuracy) and by the percentage of pages of a given class that are correctly classified as belonging to the class (coverage). Table 4.1 shows the best results obtained using the vocabulary made of the documents' original words. As expected, the class 'Other' shows a very poor performance in coverage (23.75%) due to the very diverse textual content of these documents. However, very few documents that do not belong to this class are put in this category, which gives a very high level of accuracy (93.04%) for the class 'Other'.

	Other	Department	Faculty	Staff	Student	Project	Course	Coverage
Other	147	56	34	17	169	72	123	23.75%
Department	0	25	1	1	0	0	0	92.59%
Faculty	1	2	20	0	7	3	1	58.82%
Staff	0	2	9	1	4	3	2	4.76%
Student	7	3	7	3	103	1	4	80.47%
Project	3	1	2	0	7	7	0	35.00%
Course	0	3	0	5	0	0	36	81.82%
Accuracy	93.04%	27.17%	27.40%	3.70%	35.52%	8.14%	21.69%	37.96%

Table 4.1 Results with 250-Word Vocabulary

Another interesting class is the 'Staff' class which shows very poor results as well, only 4.76% for coverage and 3.70 for accuracy. This can be explained by the fact that the textual content of these documents is very similar to the content of the 'Student' and 'Faculty' classes ('Person' super-class). From Table 4.1, it can be seen that 9 documents that belong to 'Staff' are identified as 'Faculty' and 4 as 'Student'. As a result, if measured at the level of its super-class 'Person', the coverage would be 67%. In fact, this phenomenon is the reason why the hierarchical approach in document recognition has been addressed in the following sections. The overall performance for the approach of using the original words is 37.96%. This percentage is the same for both, accuracy and coverage, so it can be used to measure performance of different types of tests.

The best performance was obtained by using the word stems instead of the original documents' words as shown on Table 4.2. As it can be seen, all the performance measures are a little better than in previous method (38.30%). However, the difference between these two tests in overall performance is less then 1%, so it is difficult to draw a conclusion about the efficiency of the tested methods.

	Other	Department	Faculty	Staff	Student	Project	Course	Coverage
Other	147	86	30	19	106	114	117	23.75%
Department	0	24	0	2	0	0	1	88.89%
Faculty	2	3	17	0	3	9	0	50.00%
Staff	3	0	7	2	5	4	0	9.52%
Student	7	7	5	0	97	5	7	75.78%
Project	2	2	1	0	1	12	2	60.00%
Course	0	1	0	0	0	0	43	97.73%
Accuracy	91.30%	19.51%	28.33%	8.70%	45.75%	8.33%	25.29%	38.30%

Table 4.2 Results with 800-Stem Vocabulary

In order to have a better idea about the performance of these two methods the Figure 4.2 shows the results obtained on the same data set, using different vocabulary sizes. For each classification method, 49 tests were performed using vocabulary sizes starting from 100, iteratively increasing by 50, up to 2500. It can be seen that using original words with a very small vocabulary gives better results than using stemmed words. However, better results are obtained with stemmed words for the majority of the tests as well as the best recorded performance among all the tests.

Besides the results, it is also the stability that differentiates these two methods. A stronger variability can be observed within the results obtained using the original words. For instance, while several results between 37% and 38% can be observed from the tests performed using stemmed words, only one of them can be seen in the other group of tests. The difference between the best result and second best result achieved with the original words is around 1% (37.96 and 36.95%), which gives the impression that it could be a matter of chance.

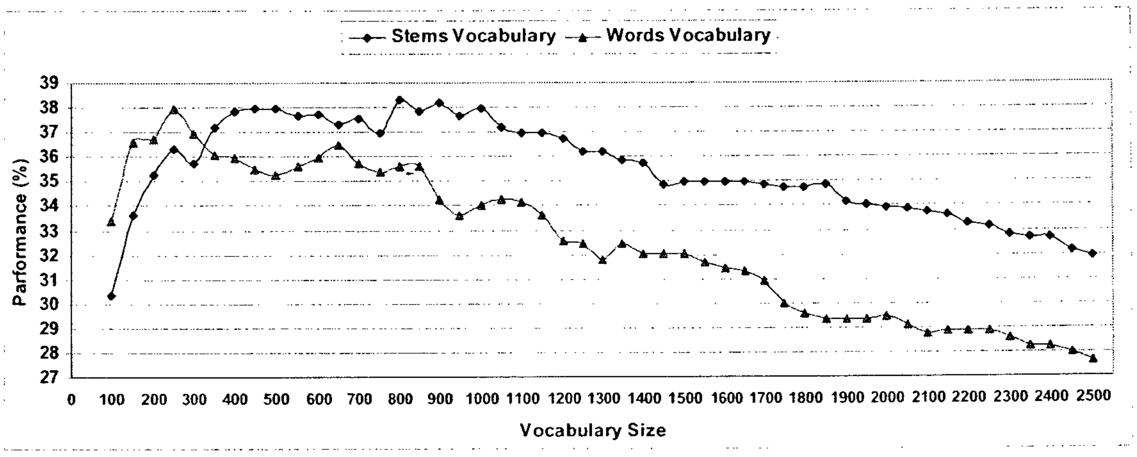


Figure 4.2 Words vs. Stems Vocabulary Performance

Because of all the characteristics of the class 'Other' discussed in this document, it is questionable how relevant it is to use these documents for the test purposes. Even the CMU text learning group have published most of their test results without this class. Figure 4.3 shows the results from the same tests as these shown on Figure 4.2 excluding the results related to the class 'Other'. It can be seen that overall performance is much higher for all the tests.

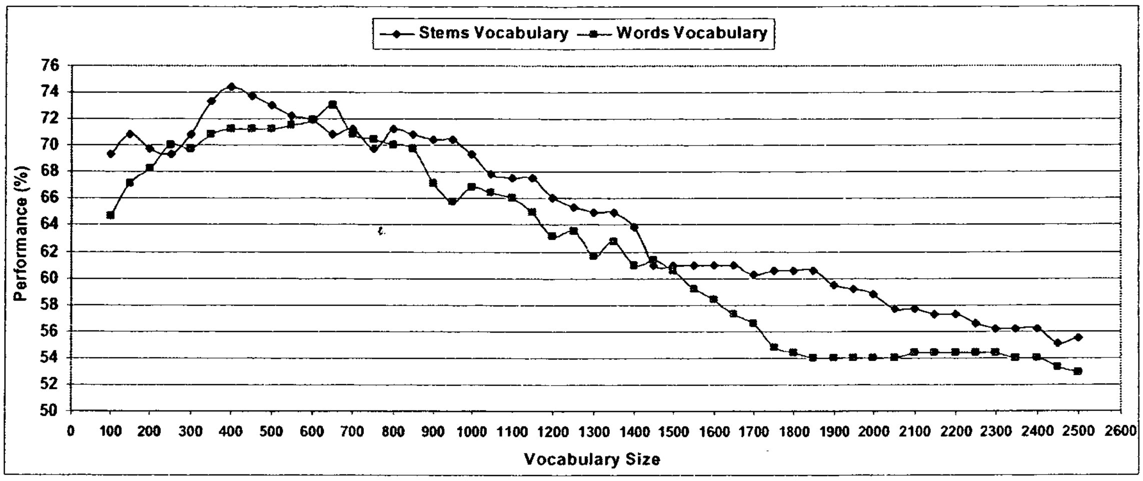


Figure 4.3 Words vs. Stems Vocabulary Performance Excluding 'Other' Class

The highest score for the method using the word stems is 74.45% whereas for the method using the original words is 72.99%. This time, the best results are achieved with the vocabulary sizes of 400 and 650 respectively, using the stems' vocabulary and the words' vocabulary. This is quite different from the previous tests where the vocabulary sizes were 800 and 250 respectively, for the stems' vocabulary and the words' vocabulary. In addition, the tendency to have better results with original words while using very small vocabularies no longer exists in these tests. The method using stemmed words shows consistently better results.

Table 4.3 and Table 4.4 show detailed information about the best results obtained using both methods. It can be seen that the performance is significantly improved when documents that belong to the class 'Other' are removed form the testing data set.

	Department	Faculty	Staff	Student	Project	Course	Coverage
Department	24	0	2	0	0	1	88.89%
Faculty	2	21	0	5	5	0	61.76%
Staff	0	8	2	6	2	3	9.52%
Student	6	6	3	102	2	4	79.69%
Project	1	1	0	2	12	1	60.00%
Course	1	0	0	0	0	43	97.73%
Accuracy	70.59%	58.33%	28.57%	88.70%	57.14%	82.69%	74.45%

Table 4.3 Results with 400-Stem Vocabulary Excluding the Class 'Other'

	Department	Faculty	Staff	Student	Project	Course	Coverage
Department	24	0	2	0	0	1	88.89%
Faculty	2	22	1	4	4	0	64.71%
Staff	1	11	2	3	1	3	9.52%
Student	4	6	5	102	0	6	79.69%
Project	2	2	1	2	9	1	45.00%
Course	1	0	2	0	0	41	93.18%
Accuracy	70.59%	53.66%	15.38%	91.89%	64.29%	78.85%	72.99%

Table 4.4 Results with 650-Word Vocabulary Excluding the Class 'Other'

4.5 Hierarchical Method

As described in this document, the approach using the hierarchy of documents performs multiple classifications of a given document. The number of classifying processes is equal (at the maximum) to the number of the ontology domains or sub-classes. The first classification is performed against the top domain made of the most upper ontology classes using the vocabulary corresponding to those classes. If a given page is classified as a class that contains sub-classes, then an additional classification is performed using a vocabulary made of these sub-classes in order to determine which one represent the best that page. This process has to be repeated until the lowest level classes are reached.

In order to see if the hierarchical approach can improve the overall results, the results of the classification at the first level must be better than final results of the non-hierarchical approach when grouped by the top level classes. Figure 4.4 shows the results obtained from both approaches.

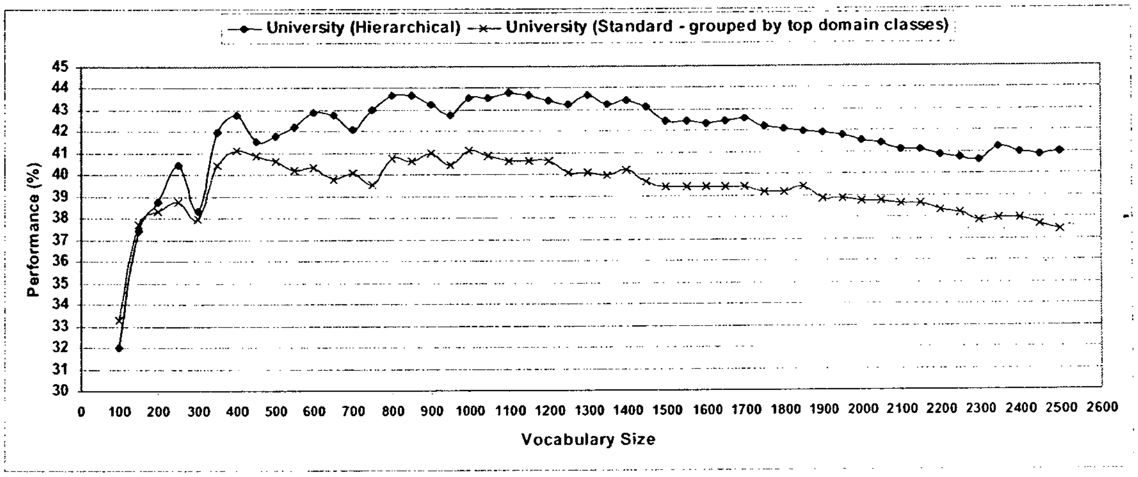


Figure 4.4 Performance of Standard vs. Hierarchical Approach

The curve representing the standard (non-hierarchical) approach is obtained from data used in Figure 4.2 representing the results of the approach using stemmed words. All the tests of the hierarchical approach are performed using stems because, as shown in this document, this method tends to improve the classification process. Once the results are grouped by the four the top domain classes ('Department', 'Person', 'Activity' and 'Other'), the curve identified as "Stems Vocabulary" on Figure 4.2 becomes the one titled "University (Standard - grouped by top domain classes)" on Figure 4.4. The other curve shown on same figure is obtained from the separate tests using the hierarchical approach. It can be seen that hierarchical approach tends to deliver better results than the standard one. Only with very small vocabulary sizes, do the results tend to be similar, but the performance achieved at that level is very poor for both approaches. Therefore these cases shouldn't be used for text classification.

In order to better understand exactly how the hierarchical approach improves the results of the classifying algorithm, Figure 4.5 shows the same results as Figure 4.2, from which the documents belonging to the class 'Other' have been removed.

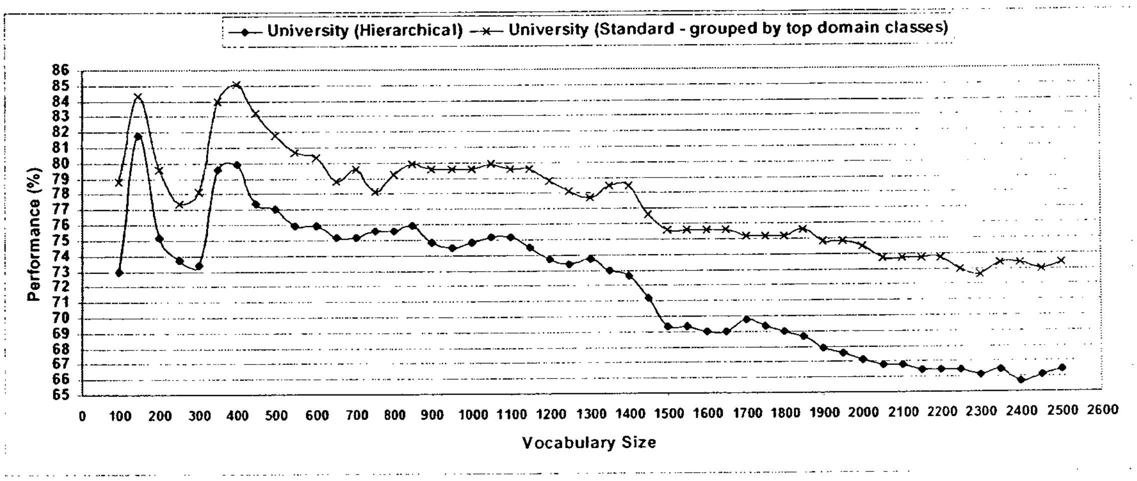


Figure 4.5 Performance of Standard vs. Hierarchical Approach Without 'Other' Class

This situation looks quite different from the previous figure. In fact, the hierarchical approach is significantly less efficient that the standard approach if the results concerning the class 'Other' are removed. The reason why the class 'Other' is advantaged to the hierarchical approach whereas other classes are disadvantaged, can be seen as an effect of the grouping of the lower level classes into higher ones. As shown in the previous section, the class 'Other' systematically obtained worse results than the other classes because of its less homogenous content. In the hierarchical approach, this characteristic of the 'Other' class is slightly neutralized because it is compared against classes with less homogenous content ('Person' and 'Activity') than was the case with classes in the standard approach ('Faculty', 'Student', 'Staff', 'Course', 'Project'). Consequently, the class 'Other' obtained better results while the success ratio has decreased for the rest of the classes.

Another interesting aspect of the hierarchical approach is the success ratio obtained among lower level classes ('Activity' and 'Person' sub-classes). It was shown that the classification at the top domain is less efficient for these classes. However, in Figure 4.6 and Figure 4.7 it can be seen that overall classification performance of these classes is rather similar.

Despite very uneven correlation between the different approaches, it is impossible to identify the best of the two approaches, neither for the 'Activity' nor for the 'Person' sub-classes. It is obvious that the gain of the overall success ratio concerning these sub-classes comes from the classifying process at the second level, which means that classifying similar classes that belong to the same domain in a separate process is more efficient then classifying them in a global process that includes all the ontology classes.

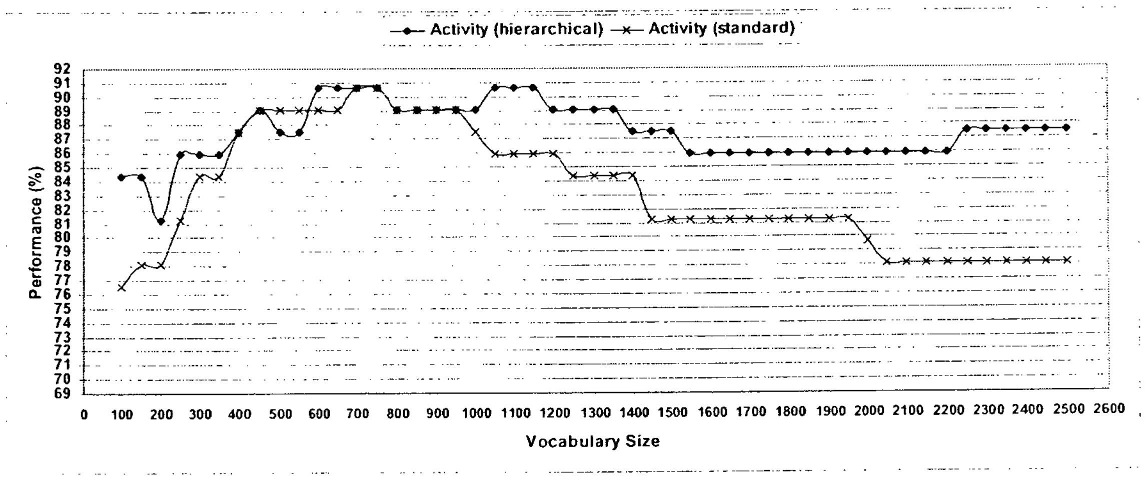


Figure 4.6 Activity – Hierarchical vs. Standard Approach

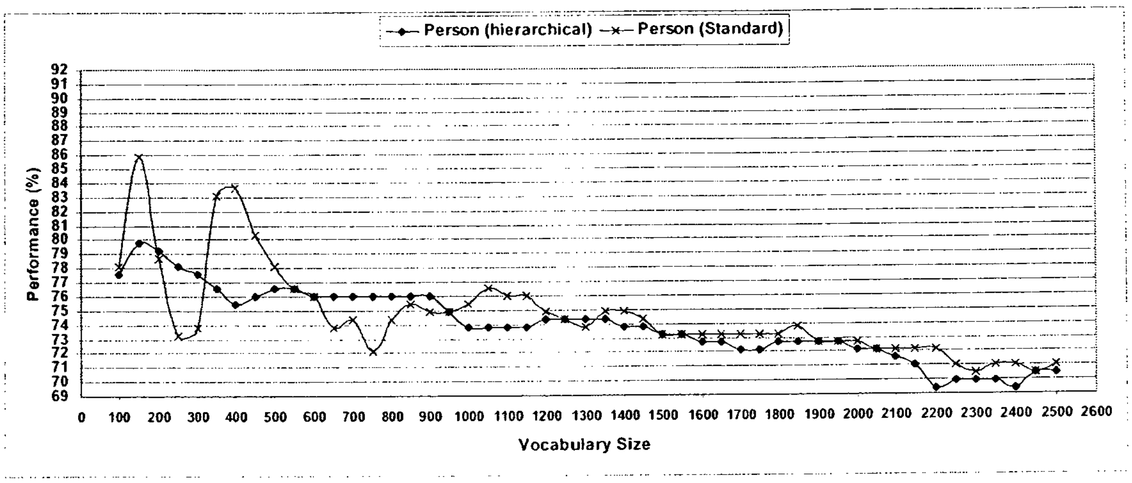


Figure 4.7 Person – Hierarchical vs. Standard Approach

One of the advantages of the hierarchical approach is the fact that vocabulary size can be different for each domain. Thus for the University ontology, the size of each vocabulary can be chosen according to the best results achieved on the training data. For instance, it can be seen from the previous figures that the best results for the top domain are obtained using 1100 features, and for the 'Activity' and 'Person' domains using respectively 600 and 150 features. When performed using these vocabulary sizes, the overall performance for the Ontology class is 40.65% which is slightly higher than the score obtained using the standard approach (38.30%). The detailed information regarding the scores for each class is shown on Table 4.5.

	Other	Department	Faculty	Staff	Student	Project	Course	Coverage
Other	186	111	28	0	94	84	116	30.05%
Department	0	24	1	0	0	0	2	88.89%
Faculty	1	4	21	0	3	3	2	61.76%
Staff	2	0	12	0	4	3	0	0.00%
Student	12	11	8	0	82	4	11	64.06%
Project	3	4	2	0	0	10	1	50.00%
Course	0	3	0	0	0	1	40	90.91%
Accuracy	91.18%	15.29%	29.17%	0.00%	44.81%	9.52%	23.26%	40.65%

Table 4.5 Hierarchical Approach Results

Table 4.6 shows that the best results were obtained when documents that belong to 'Other' class are excluded. For this test, vocabulary sizes for 'Activity' and 'Person' have been kept the same as in the previous test (600 and 150 respectively) while the size of the top domain has been adjusted from 1100 to 150. This vocabulary size provides the best results for the top domain as it is shown on Figure 4.5.

	Department	Faculty	Staff	Student	Project	Course	Coverage
Department	24	2	0	0	0	1	88.89%
Faculty	3	25	0	5	1	0	73.53%
Staff	2	10	0	6	2	0	0.00%
Student	4	7	0	100	5	4	78.13%
Project	1	4	0	6	6	1	30.00%
Course	4	0	0	0	1	39	88.64%
Accuracy	63.16%	52.08%	0.00%	85.47%	40.00%	86.67%	70.80%

Table 4.6 Hierarchical Approach Results without 'Other' Class

As expected, the results are not as good as those obtained using the standard method. The overall performance in the hierarchical approach was 70.80% whereas in the standard approach, using stemmed words, the total success ratio was 74.45%.

4.6 Mixed Method

The analysis that has been introduced so far in this chapter is related to the Standard and Hierarchical methods. In summary, the Standard method tends to achieve better results when the class 'Other' is excluded, which indicates that this approach is more suitable to the classes that contain documents with homogenous content. On the other hand, Hierarchical method performs better with the data set including the class 'Other'. The opposite behaviour of this method shows its ability to deal better with the classes that contain documents less similar to each other. The purpose of the Mixed method, as explained in Chapter 2, is to merge the best of each method into a single method. The results shown in Table 4.7 were obtained using the Mixed method with a 400-word vocabulary size for the University ontology that is used at the first classifying process and 600 and 150 respectively for 'Activity' and 'Person' domain vocabularies. While the vocabulary sizes for 'Activity' and 'Person' domains are the same as in the Hierarchical method (since the classification processes are the same at that level), the size of the first vocabulary is different form the one used in the previous approach. It corresponds to the vocabulary size that produces the best results, grouped by the top level classes, using the Standard approach. This is the most optimal size for this vocabulary because the classification in this step was achieved by using the standard approach and then the obtained results were grouped by the top level classes. For both types of tests, including and excluding the class 'Other', the size of this vocabulary size was 400. It can be seen on Figure 4.4 as well as on Figure 4.5 that this size is optimal in each case.

Table 4.7 shows the results obtained using this method including the class 'Other'. The overall success ratio is 37.63% which is worse then either the Standard (38.30%) or the Hierarchical (40.65) approach. Table 4.8 illustrates the results for the data set excluding the class 'Other'. This time, the overall performance is 73.36%, which is between the Hierarchical (70.80%) and the Standard (74.45%) methods.

	Other	Department	Faculty	Staff	Student	Project	Course	Coverage
Other	135	42	33	0	169	110	130	0.2181
Department	0	24	1	0	1	0	1	88.89%
Faculty	1	2	23	0	3	4	1	67.65%
Staff	0	0	12	0	4	3	2	0.00%
Student	5	6	10	0	101	2	4	78.91%
Project	3	1	1	0	2	12	1	60.00%
Course	0	1	0	0	0	2	41	93.18%
Accuracy	93.75%	31.58%	28.75%	0.00%	36.07%	9.02%	22.78%	37.63%

Table 4.7 Mixed Approach Results

	Department	Faculty	Staff	Student	Project	Course	Coverage
Department	24	1	0	1	0	1	88.89%
Faculty	2	23	0	3	4	1	67.65%
Staff	0	12	0	4	3	2	0.00%
Student	6	10	0	101	2	4	78.91%
Project	1	1	0	2	12	1	60.00%
Course	1	0	0	0	2	41	93.18%
Accuracy	70.59%	48.94%	0.00%	90.99%	52.17%	82.00%	73.36%

Table 4.8 Mixed Approach Results without Other 'Class'

From the results presented in these tables it can be seen that combining more efficient features from different methods doesn't necessarily assure better results. It can also be noticed that while in some case overall performance increases in hierarchical and mixed methods, the performance regarding the class 'Staff' is in all cases equal to 0%. The problems related to this class have already been addressed in previous sections. Clearly, neither Hierarchical nor Mixed method is able to improve the results concerning this class. On the contrary, even a very low success ratio from the Standard method was impossible to reach. Although no additional practical methods are introduced in this document, the theoretical aspects of this problem are further addressed in the following sections.

Chapter 5

Conclusion

5.1 Conclusion

Annotating existing Web documents for the Semantic Web is a very audacious objective that has to be completed in order to make the Semantic Web a reality. At present, automatic and semi-automatic methods are the only credible approaches to achieve this task even though they cannot yet provide an acceptable level of accuracy. As demonstrated in this work, as well as in several other similar works, it is possible to reach a relatively high ratio of success in recognizing instances of some ontology classes whereas some others obtain a rather small degree of success. In some cases, it is possible to improve the accuracy of classifying algorithms by adjusting some of the parameters or by changing the algorithm itself. While some other works ([17], [18]) have maintained the focus on comparing the efficiency of different classifying algorithms, in this work the main objective was to identify and to quantify the impact of some other variables that can also impact significantly results of an algorithm for classifying Web documents. Thus, all the tests in this work have been performed using the same algorithm, based on Naive Bayes text classifier. On the other hand, aspects such as feature selection and hierarchy of documents are put forward as potential candidates to improve the process of classifying Web documents.

Feature selection has a very strong impact on the efficiency of document classification algorithms. Various tests performed in this work show that the accuracy rate of an algorithm can fluctuate significantly when the number of selected features representing a domain (domain vocabulary size) is changed. For instance, in the case of tests performed using stemmed words shown on Figure 4.2, it can be seen that when the vocabulary size is increased from 700 to 2500, the success ratio drops by more than 6%. Therefore, it is very important to find the most optimal vocabulary size. In this work, the optimal size for each test was found by using an experimental method, by performing the tests using several different vocabulary sizes and then choosing the size that allows the best results. In all the tests, the success ratio tends to follow a Poison distribution while increasing the vocabulary size, which means that theoretically there is only one optimal size. In the real world, it could be difficult to find the optimal number of features for a domain. In the case of the university ontology used in this work, there are around 48000 features. Even by eliminating unique words and by converting them into their stems the number of features remains relatively high (16000). Knowing that vocabulary sizes follow a Poison distribution at least indicates that the optimal size should be relatively small. However, as demonstrated in this work, results can be significantly different even with small vocabulary size changes. One of the possible solutions to this problem could possibly be found by analysing the frequency of the features in the training data set. Figure 4.1 shows the number of documents represented by different numbers of features (stemmed words) in the training data set used for the university ontology. For instance, it can be seen that there are only around 500 features that can be found in approximately 400 documents. By increasing the number of features, the number of documents that contain these features gets smaller and this behaviour follows an exponential distribution. In the case of the university ontology, the mean of that distribution is 715, which is very close to the vocabulary size of 800 that was experimentally found to be the optimal size for the same training data set. If this hypothesis is true, the difference between the two vocabulary sizes could be explained by the fact that the training data used to calculate the mean is different from the testing data that was used to experimentally find the optimal size. Because of the limited availability of testing data, no additional tests have been performed to confirm or to reject this hypothesis.

Using word stems instead of original words was another aspect of feature selection that was introduced in this work. Word stems bring several advantages over original words in text classification. First of all, the total number of features is reduced by using word stems. In the case of the university ontology, the total number of features is reduced from around 63000 to approximately 48000, which is very important for software performance. The second advantage of using word stems is the increase of the success ratio. In all the tests with optimal vocabulary sizes, the best results were achieved using word stems. Although, not very significant, the results were approximately 1% better than identical tests with original words. Finally, the third advantage that was observed during the tests is the consistency of the results. With word stems, better results are achieved not only with the optimal vocabulary sizes, but also with almost all other vocabulary sizes. Moreover, the results tend to decrease by a smaller degree when using non-optimal vocabulary sizes which can be a very interesting characteristic knowing that finding optimal vocabulary size can be a difficult task.

The hierarchy of documents is another aspect that was introduced and evaluated in the context of text classification. From the test results, it can be seen that not all types of ontology classes benefit from this approach. In the case of the university ontology, the results were improved mainly within the documents of the class 'Other'. The reason for that is the fact that this class contains documents with a very different textual content and the hierarchical approach tends to privilege this type of classes. Although, the overall results were improved with this method, it can be seen that in fact the results per class actually decreased for almost all the classes except the class 'Other'. Therefore, depending on importance of the ontology classes, this method might or might not be advised.

In the hierarchical approach, an ontology is in fact divided into several other ontologies or domains and then each of them is used to perform a separate classification of the corresponding documents. This approach brings the possibility of fine tuning each document classification process separately which might be potentially advantageous for very large ontologies. For instance, for each domain it is possible to create separate feature selections and determine optimal vocabulary sizes for each of them.

5.2 Future Work

As mentioned in this document, there are still a lot of Semantic Web areas that need to be improved. The particularity of automatic and semi-automatic Semantic Web annotations is the fact that it is impossible to make a 100% accurate solution. Even if it was possible, such a situation would not last long since the Web is constantly evolving and it should be readjusted periodically which means that some defects are always unavoidable. From the tests performed in this work, it can be seen that there are a lot of different ways to improve the accuracy of a classification algorithm. Adjusting the vocabulary size, using a hierarchy of documents, or refining the feature selection process are just a few of the numerous possibilities to improve the results. However, no matter which approach is used; some documents cannot be properly classified because their textual content is different from those used for the training data. Such situations can occur for various reasons. Some documents can simply use other words to describe the same concepts which sometimes can be improved by adding more documents to the training data. Another approach could be creating a more intelligent feature selection process. It was shown that using stemmed words instead of the original words improves the results. This was due to the fact that some documents are wrongly eliminated, because they use a different writing style. The feature selection could be further improved by adding synonyms to some selected features or by detecting them in documents that have to be classified and converting them into selected features.

Besides the semantic reasons for misclassifying documents, there are also defects caused by the structure of the ontologies. Most of the ontologies are developed by humans according to their knowledge about the domain that has to be described. The problem is that ontologies are meant to be used by machines. Therefore, the ontology classes might not represent the reality that is perceived by computers. For instance, the ontology class 'Other' used in this work is in fact a potpourri of several other classes. It is obvious that such a class cannot have a high degree of success in a document classification. To a different degree, the same remark can be made for any other class. From the perspective of an automatic classification process made by computers, it could be more efficient to subdivide existing classes into several smaller classes that do not necessarily have a particular significance for humans. For example, the class Department could be broken into sub-classes such as 'Department with less then 100 words', 'Department with more then 100 words', 'Department with parent page menu', 'Department with sub-pages', and so forth. It would be also interesting to compare results between the hierarchical and standard approach with such divided classes.

Recognizing relations between extracted class instances was not a topic in this work. However that subject has the same importance as the semantic annotation of Web documents. The complete picture of the Semantic Web must include the relations between the ontology classes in order to allow computers to read and reason about various Web resources. Logically, recognizing class instances comes first and then finding relations between them second. However, knowing the relations that a class can have with others might be useful for the process of class identification itself. For instance, knowing that a course is taught by a professor could possibly facilitate identifying instances of the class 'Professor' once an instance of the class 'Course' is discovered.

All the topics and issues mentioned in this section for future works are very complex subjects and knowing the scale of the Web it is impossible to expect a single software solution to all of them. The most realistic way to handle these types of problems is by using distributed multi-agent systems. Semantic annotation of Web documents depends on several areas that can be implemented through different agents: creation and evolution of ontologies, process of recognizing (classifying) Web documents, process of recognizing relations between classes, control and verification of Semantic Web annotations. Each of the identified groups can be contain multiple agents specialized by domain or geography. Probably the most challenging is the control and verification of Semantic Web annotations, because it has to be done ultimately by humans. Only the end-users of the Semantic Web are actually in position to evaluate the success ratio of a Semantic Web annotation process. With their feedback, it would be possible not only to re-annotate properly the misclassified documents but also to adjust any parameter used in the process of the semantic annotation. Moreover, this information could be used to modify existing ontologies and create new ones. At this stage of Semantic Web development, it is not yet obvious how such a feedback could be obtained from the end-users. However, some other areas could be considered as an example of handling similar issues. For instance, in the e-mail spam detecting software, the option of spam releasing could be a good example of providing the end-user feedback on an automatic process of another type of document classification process.

Multi-agent systems are closely tied with the Semantic Web. The ontologies represent a very important piece in the artificial reasoning that can enable various intelligent agents to communicate and cooperate using worldwide Web resources. Semantic Web services are the perfect example of such an initiative. Multi-agent systems will not only benefit the most from the Semantic Web, but also be used to build and maintain the next generation of the internet.

References

[1] Jorge Cardoso and Amit P. Sheth, Semantic Web Services, Processes and Applications, August 3, 2006

[2] Liyang Yu, Introduction to the Semantic Web and Semantic Web Services, June 14, 2007

[3] WSDL Web Services Description Language, http://www.w3.org/TR/wsdl, W3C Note March 15, 2001

[4] UDDI Spec Technical Committee, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, October 19, 2004

[5] RDF/XML Syntax Specification, http://www.w3.org/TR/rdf-syntax-grammar, W3C Recommendation February 10, 2004

[6] RDF Vocabulary Description Language 1.0: RDF Schema, http://www.w3.org/TR/rdf-schema, W3C Recommendation February 10, 2004

[7] OWL Web Ontology Language, http://www.w3.org/TR/owl-features, W3C Recommendation February 10, 2004

[8] OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S, W3C Member Submission, November 22, 2004

[9] WSDL-S Web Service Semantics, http://www.w3.org/Submission/WSDL-S, W3C Member Submission November 7, 2005

[10] WSMO, Web Service Modeling Ontology, http://www.w3.org/Submission/WSMO/, W3C Member Submission, June 3, 2005

[11] Daniel Boley, Maria Gini, Robert Gross, Eui-Hong (Sam) Han, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher and Jerome Moore, Partitioning-Based Clustering for Web Document Categorization, Decision Support Systems, Vol. 27, No. 3. (1999), pp. 329-341.

[12] Becket D. Modernising Semantic Web Markup, 2004 http://www.idealliance.org/papers/dx_xmle04/papers/03-08-03/03-08-03.html#problems

[13] Ying Ding, Schubert Foo, Ontology Research and Development Part 1, 2002

[14] Nadzeya Kiyavitskaya, Nicola Zeni, James R. Cordy, Luisa Mich and John Mylopoulos, pp.14 – 15, Semi-Automatic Semantic Annotations for Web Documents, Proc. SWAP 2005, 2nd Italian Semantic Web Workshop, 2005.

[15] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam and Sean Slattery, Learning to Construct Knowledge Bases from the World Wide Web, http://www.cs.cmu.edu/~webkb/, pp. 69 – 113, Artif. Intell, 2000

[16] The Porter Stemming Algorithm, http://tartarus.org/~martin/PorterStemmer/

[17] Elena Montañés, Irene Díaz, José Ranilla, Elías F. Combarro, and Javier Fernández, University of Oviedo, Scoring and Selecting Terms for Text Categorization

[18] Dunja Mladenic, Marko Grobelnik, Feature selection on hierarchy of Web documents

[19] Marilyn R. Wulfekuhler, William F. Punch, Finding Salient Features for Personal Web Page Categories

[20] Bo Leuf, The Semantic Web – Crafting Infrastructure for Agency

[21] John Davies, Rudi Studer, Paul Warren, Semantic Web Technologies – Trends and Research in Ontology-based Systems

[22] D. D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In Third Annual Symposium on Document Analysis and Information Retrieval, pages 81 – 93, 1994.

[23] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. Improving text classification by shrinkage in a hierarchy of classes. In Proceedings of the 15th International Conference on Machine Learning, pages 359 – 367. Morgan Kaufmann, 1998.

[24] Sean Luke, Lee Spector, David Rager and James Hendler, Ontology Web Based Agents

[25] Marko Balabanovic, Yoav Shoham, Yeogirl Yun, An Adaptive Agent for Automated Web Browsing

[26] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning, 29:103 – 130, 1997.

[27] A. K. McCallum and K. Nigam. A comparison of event models for Naive Bayes text classification. In Working Notes of the ICML/AAAI Workshop on Learning for Text Categorization, 1998. http://www.cs.cmu.edu/~mccallum.

[28] Stephen Soderland. Learning information extraction rules for semi-structured and free text. Machine Learning, 34(1), February 1999.

[29] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In Proceedings of the Fourteenth International Conference on Machine Learning, pages 143 – 151, Nashville, TN, 1997. Morgan Kaufmann.

[30] D. Koller and M. Sahami. Toward optimal feature selection. In Proceedings of Thirteenth International Conference on Machine Learning. Morgan Kaufmann, 1996.

[31] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In Proceedings of the Fourteenth International Conference on Machine Learning, pages 170 – 178, Nashville, TN, 1997. Morgan Kaufmann.

[32] G.H. John, R. Kohavi, K. Pfleger, Irrelevant features and the subset selection problem, Proceedings of the 11th International Conference on Machine Learning ICML94, San Francisco, CA, 1994, pp. 121– 129. , Morgan Kaufmann.

[33] Cimiano, P., Handschuh, S., and Staab, S.: Towards the self-annotating web. In Proceedings of the 13th international conference on World Wide Web,462–471, ACM Press, 2004

[34] Decker, S., Erdmann,M., Fensel,D., and Studer, R.: Ontobroker: Ontology based access to distributed and semi-structured unformation. In DS-8: Database Semantics - Semantic Issues in Multimedia Systems, IFIPTC2/WG2.6 Eighth Working Conference on Database Semantics, 351–369, Rotorua, New Zealand, 1999

[35] Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo,T., McCurley, K. S., Rajagopalan, S., Tomkins, A., Tomlin, J.A., Zien, J. Y.: A Case for Automated Large-Scale Semantic Annotation. Journal of Web Semantics, 1(1) 115–132, 2003

[36] Etzioni, O., Cafarella, M.J., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Unsupervised named-entity extraction from the web: An experimental study. Artificial Intelligence 165, 91 – 134, 2005

[37] Yang, Y, Noise Reduction in a Statistical Approach to Text Categorization, Proc. of SIGIR'95, pp. 256-263, 1995

[38] Handschuh, S., Staab, S., Ciravegna, F.: S-CREAM- Semi-automatic Creation of Metadata. The 13th International Conference on Knowledge Engineering and Management (EKAW2002), ed. Gomez-Perez, A., Springer Verlag, 2002

[39] Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic Annotation, Indexing, and Retrieval. Elsevier's Journal of Web Sematics, 2(1), 2005

[40] Leidner, J. L.: Current Issues in Software Engineering for Natural Language Processing. Proc. of the Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS), the Joint Conf. for Human Language Technology and the Annual Meeting of the Noth American Chapter of the Association for Computational Linguistics (HLT/NAACL'03), Edmonton, Alberta, Canada, 45–50

[41] Kogut, P. and Holmes,W.: AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. First International Conference on Knowledge Capture (K-CAP 2001). Workshop on Knowledge Markup and Semantic Annotation, Victoria, B.C., Canada, October 2001

[42] Muslea, I., Minton, S., Knoblock, C.A.: Active learning with strong and weak views: A case study on wrapper induction. In Proc. 18th Int. Joint Conference on Artificial Intelligence, 415-420, 2003

[43] Nobata, C., Sekine, S.: Towards automatic acquisition of patterns for information extraction. In Proc. International Conference on Computer Processing of Oriental Languages, 1999

[44] Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A., Goranov, M.: Towards Semantic Web Information Extraction. Human Language Technologies Workshop at the 2nd International Semantic Web Conference (ISWC2003), 20 October 2003, Florida, USA

[45] Yang, Y. & Liu, X, A Re-examination of Text Categorization Methods, Proc. of SIGIR'99, pp. 42-49, 1999

[46] Torkkola, K, Linear Discriminant Analysis in Document Classification, 2002

[47] Sebastiani, F, Machine Learning in Automated Text Categorization, ACM Computing Surveys, 34(1):1-47, 2002

[48] Bettina Berendt, Andreas Hotho, and Gerd Stumme, Semantic Web Mining and the Representation, Analysis, and Evolution of Web Space, In Proceedings of RAWS 2005 Workshop

[49] D. Mladenic and M. Grobelnik, "Feature Selection for Unbalanced Class Distribution and Naive Bayes," Proc. 16th Int'l Conf. Machine Learning (ICML 99), Morgan Kaufmann, 1999, pp. 258–267.

[50] Y. Yang and J.O. Pedersen, A Comparative Study on Feature Selection in Text Categorization, Proc. 14th Int'l Conf. Machine Learning (ICML 97), Morgan Kaufmann, 1997, pp. 412–420.

[51] Daniel Boley, Maria Gini, Robert Gross, Eui-Hong Han, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher and Jerome Moore, Document Categorization and Query Generation on the World Wide Web Using WebACE, pp. 365—391, vol. 13, AI Review.

[52] Tao Liu Zheng Chen, Benyu Zhang, Wei-ying Ma, Gongyi Wu, Improving Text Classification using Local Latent Semantic Indexing, pp.162-169, Fourth IEEE International Conference on Data Mining (ICDM'04), 2004

[53] WebACE: a Web agent for document categorization and exploration, Eui-Hong (Sam) Han, Daniel Bole, Maria Gin, Robert Gross, Kyle Hastings, George Karypis, Vipin Kuma, Bamshad Mobasher, Jerome Moore, 1998

[54] C.C. Aggarwal, F. Al-Garawi, and P.S. Yu. Intelligent crawling on the World Wide Web with arbitrary predicates. In Proceedings of the WWW Conference, 2001.

[55] B. Berendt, A. Hotho, and G. Stumme. Usage mining for and on the semantic web, In [57], pages 461-480. AAAI/MIT Press, 2004.

[56] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic specific Web resource discovery. Computer Networks, 31:1623 - 1640, 1999.

[57] A. Doan, R. McCann, and W. Shen. Collaborative development of information integration systems. In [26], pages 34 - 41. 2005.

[58] A. Kralisch and B. Berendt. Language-sensitive search behaviour and the role of domain knowledge. New Review in Hypermedia and Multimedia, Vol. 11, No. 2. pp. 221-246, 2005

[59] Borislav Popov, Atanas Kiryakov, Angel Kirilov, Dimitar Manov, Damyan Ognyanoff, Miroslav Goranov, KIM – Semantic Annotation Platform, 2nd International Semantic Web Conference (ISWC2003), 20-23 October 2003, Florida, USA. LNAI Vol. 2870, pp. 834-849, Springer-Verlag Berlin Heidelberg 2003.

Appendix

Source Code 1: evaluate_document_class

```
CREATE   FUNCTION   evaluate_document_class(_IdWebRessource   int(10)   unsigned)
RETURNS int(10) unsigned
BEGIN
  RETURN (
    SELECT IdOntologyClass
    FROM (
      SELECT                                                 e.IdOntologyClass,
(LOG(g.Probability)/f.WordsCount)+SUM(((d.Occurrence/f.WordsCount)*LOG(e.Probab
ility/(d.Occurrence/f.WordsCount)))) Ranking
      FROM  feature_selection  c,  word_occurrence  d,  class_word_probability  e,
web_ressource f, class_probability g
      WHERE d.IdWebRessource = _IdWebRessource
      AND d.IdWebRessource = f.IdWebRessource
      AND c.Word = d.Word
      AND c.Word = e.Word
      AND c.idDomain = 1
      AND c.idDomain = e.idDomain
      AND e.idDomain = g.idDomain
      AND e.IdOntologyClass = g.IdOntologyClass
      GROUP BY e.IdOntologyClass
      ORDER BY 2 DESC ) AS a
    LIMIT 1);
END;
```

Source Code 2: evaluate_document_class_hier

```
CREATE  FUNCTION  evaluate_document_class_hier(_IdWebRessource  int(10)  unsigned)
RETURNS int(10) unsigned
BEGIN
  DECLARE _Ret int(10) unsigned;
  DECLARE _IdDomain int DEFAULT 2;

  WHILE (_IdDomain > 1) DO
    SELECT IdOntologyClass
    INTO _Ret
    FROM (
      SELECT IdOntologyClass
      FROM (
        SELECT                                               e.IdOntologyClass,
(LOG(g.Probability)/f.WordsCount)+SUM(((d.Occurrence/f.WordsCount)*LOG(e.Probab
ility/(d.Occurrence/f.WordsCount)))) Ranking
        FROM  feature_selection  c,  word_occurrence  d,  class_word_probability  e,
web_ressource f, class_probability g
        WHERE d.IdWebRessource = _IdWebRessource
        AND d.IdWebRessource = f.IdWebRessource
        AND c.Word = d.Word
        AND c.Word = e.Word
        AND c.IdDomain = _IdDomain
        AND c.IdDomain = e.IdDomain
        AND e.IdDomain = g.IdDomain
        AND e.IdOntologyClass = g.IdOntologyClass
        GROUP BY e.IdOntologyClass
        ORDER BY 2 DESC ) AS a
      UNION ALL
      SELECT 4) AS b
    LIMIT 1;

    SELECT IdDomain
    INTO _IdDomain
    FROM (
      SELECT IdDomain
      FROM domain
      WHERE IdOntologyClass = _Ret
      UNION ALL
      SELECT 0) a
    LIMIT 1;
  END WHILE;
  RETURN _Ret;
END;
```

Source Code 3: evaluate_document_class_mix

```
CREATE  FUNCTION  evaluate_document_class_mix(_IdWebRessource  int(10)  unsigned)
RETURNS int(10) unsigned
BEGIN
  DECLARE _IdDomain int DEFAULT 1;
  DECLARE _ToRefine int DEFAULT 0;
  DECLARE _Ret int(10) unsigned;

  WHILE (_ToRefine < 2) DO
    SELECT IdOntologyClass
    INTO _Ret
    FROM (
      SELECT IdOntologyClass
      FROM (
        SELECT                                                     e.IdOntologyClass,
(LOG(g.Probability)/f.WordsCount)+SUM(((d.Occurrence/f.WordsCount)*LOG(e.Probab
ility/(d.Occurrence/f.WordsCount)))) Ranking
        FROM feature_selection c, word_occurrence d, class_word_probability e,
web_ressource f, class_probability g
        WHERE d.IdWebRessource = _IdWebRessource
        AND d.IdWebRessource = f.IdWebRessource
        AND c.Word = d.Word
        AND c.Word = e.Word
        AND c.IdDomain = _IdDomain
        AND c.IdDomain = e.IdDomain
        AND e.IdDomain = g.IdDomain
        AND e.IdOntologyClass = g.IdOntologyClass
        GROUP BY e.IdOntologyClass
        ORDER BY 2 DESC ) AS a
      UNION ALL
      SELECT 4) AS b
    LIMIT 1;

    SELECT IdDomain
    INTO _IdDomain
    FROM (
      SELECT b.IdDomain
      FROM ontology_class a, domain b
      WHERE a.IdParentOntologyClass = b.IdOntologyClass
      AND a.IdParentOntologyClass != 1
      AND a.IdOntologyClass = _Ret
      UNION ALL
      SELECT 0) a
    LIMIT 1;
    IF (_IdDomain > 0) THEN
      SET _ToRefine = _ToRefine + 1;
    ELSE
      SET _ToRefine = 2;
    END IF;
  END WHILE;
  RETURN _Ret;
END;
```

Source Code 4: extract_web_ressource_words

```
CREATE PROCEDURE extract_web_ressource_words(_IdWebRessource int(10) unsigned,
_uniqueCall bit, _StemWord bit)
BEGIN
  DECLARE TextLength INT;
  DECLARE Next INT;
  DECLARE Previous INT;
  DECLARE Text VARCHAR(50000);
  DECLARE Done INT DEFAULT 0;
  DECLARE CurWebRessourceContent CURSOR FOR
    SELECT  CONCAT(LTRIM(REPLACE(REPLACE(REPLACE(REPLACE(Content,  CHAR(13)  ,'
'),'/',' '),'—',' '),'-',' ')),' ')
    FROM web_ressource_content
    WHERE IdWebRessource = _IdWebRessource;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
  IF (_StemWord IS NULL) THEN
    SELECT CAST(Value AS UNSIGNED)
    INTO _StemWord
    FROM software_settings
    WHERE IdSoftwareSetting = 1;
  END IF;
  IF (_uniqueCall = 1) THEN
    DROP TEMPORARY TABLE IF EXISTS WebRessourceWords;
    CREATE TEMPORARY TABLE WebRessourceWords(Word VARCHAR(255));
  ELSE
    TRUNCATE TABLE WebRessourceWords;
  END IF;
  OPEN CurWebRessourceContent;
    REPEAT
      FETCH CurWebRessourceContent INTO Text;
      IF NOT done THEN
        SET TextLength = LENGTH(Text);
        SET Previous = 1;
        WHILE Previous < TextLength DO
          SET Next = LOCATE(' ', Text, (Previous));
          INSERT            INTO            WebRessourceWords            VALUES
(clean_word(LOWER(SUBSTRING(SUBSTRING(Text,Previous,Next-
Previous),1,255)),_StemWord));
          SET Previous = Next+1;
        END WHILE;
      END IF;
    UNTIL done END REPEAT;
  CLOSE CurWebRessourceContent;

  IF (_uniqueCall = 1) THEN
    DELETE
    FROM word_occurrence
    WHERE IdWebRessource = _IdWebRessource;
  END IF;
```

```
    INSERT INTO word_occurrence (IdWebRessource, Word, Occurrence)
    SELECT _IdWebRessource, a.Word, COUNT(0)
    FROM WebRessourceWords a
    WHERE a.Word != ' '
    AND NOT EXISTS (
      SELECT 0
      FROM stop_word b
      WHERE b.Word = a.Word
      AND b.IdOntologyClass IS NULL
    )
    GROUP BY a.Word;

    UPDATE web_ressource x
    SET x.WordsCount = (
      SELECT SUM(y.Occurrence)
      FROM word_occurrence y
      WHERE y.IdWebRessource = x.IdWebRessource)
    WHERE x.IdWebRessource = _IdWebRessource;
  END;
```

Source Code 5: extract_all_web_ressource_words

```
CREATE PROCEDURE extract_all_web_ressource_words()
BEGIN
  DECLARE _IdWebRessource INT(10) UNSIGNED;
  DECLARE Done INT DEFAULT 0;
  DECLARE CurWebRessource CURSOR FOR
    SELECT IdWebRessource
    FROM web_ressource;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
  TRUNCATE TABLE word_occurrence;
  DROP TEMPORARY TABLE IF EXISTS WebRessourceWords;
  CREATE TEMPORARY TABLE WebRessourceWords(Word VARCHAR(255));
  CREATE INDEX WebRessourceWordsInd1 ON WebRessourceWords(Word);
  OPEN CurWebRessource;
    REPEAT
      FETCH CurWebRessource INTO _IdWebRessource;
      IF NOT done THEN
        CALL extract_web_ressource_words(_IdWebRessource, 0, (
          SELECT CAST(Value AS UNSIGNED)
          FROM software_settings
          WHERE IdSoftwareSetting = 1));
      END IF;
    UNTIL done END REPEAT;
  CLOSE CurWebRessource;
  DROP TEMPORARY TABLE IF EXISTS WebRessourceWords;
  UPDATE software_settings
  SET Value = (SELECT CURDATE())
  WHERE IdSoftwareSetting = 2;
END;
```

Source Code 6: extract_trn_word_occurrence

```
CREATE PROCEDURE extract_trn_word_occurrence()
BEGIN

  TRUNCATE TABLE trn_word_occurrence;
  INSERT INTO trn_word_occurrence
  SELECT   a.IdWordOccurrence,   a.IdWebRessource,   b.IdOntologyClass,   a.Word,
a.Occurrence
  FROM word_occurrence a, class_instance b
  WHERE a.idWebRessource = b.idWebRessource
  AND b.Confirmed = 1;

  DROP TEMPORARY TABLE IF EXISTS trn_word_occurrence_temp;
  CREATE TEMPORARY TABLE trn_word_occurrence_temp as
  SELECT *
  FROM trn_word_occurrence a
  WHERE EXISTS(
    SELECT 'x'
    FROM trn_word_occurrence b
    WHERE b.Word = a.Word
    AND b.IdTrnWordOccurrence != a.IdTrnWordOccurrence);

  TRUNCATE TABLE trn_word_occurrence;

  INSERT INTO trn_word_occurrence
  SELECT *
  FROM trn_word_occurrence_temp;

  DROP TEMPORARY TABLE IF EXISTS trn_word_occurrence_temp;

  UPDATE web_ressource x
  SET x.WordsCount = (
    SELECT SUM(y.Occurrence)
    FROM trn_word_occurrence y
    WHERE y.IdWebRessource = x.IdWebRessource)
  WHERE EXISTS(
    SELECT 'x'
    FROM trn_word_occurrence z
    WHERE z.IdWebRessource = x.IdWebRessource
  );

  UPDATE software_settings
  SET Value = (SELECT CURDATE())
  WHERE IdSoftwareSetting = 3;
END;
```

Source Code 7: load_feature_selection

```
CREATE PROCEDURE load_feature_selection()
BEGIN

  DROP TABLE IF EXISTS ontology_class_members_temp;
  CREATE TABLE ontology_class_members_temp AS
  SELECT CASE WHEN b.idOntologyClass IS NULL THEN 0 ELSE a.idOntologyClass END
idDomain,
    IFNULL(b.idOntologyClass, a.idOntologyClass) idOntologyClass,
    IFNULL(IFNULL(c.idOntologyClass,   b.idOntologyClass),   a.idOntologyClass)
idConnectingOntologyClass
  FROM ontology_class AS a
  LEFT JOIN ontology_class AS b ON b.idParentOntologyClass = a.idOntologyClass
  LEFT JOIN ontology_class AS c ON c.idParentOntologyClass = b.idOntologyClass
  ORDER BY 1, 2, 3;

  UPDATE ontology_class_members_temp b
  SET b.idDomain = (
    SELECT a.idDomain
    FROM domain a
    WHERE (b.idDomain = a.idOntologyClass
    AND a.Hierarchical = 1)
      OR (b.idDomain = 0
      AND a.Hierarchical = 0));

  TRUNCATE TABLE class_probability;
  INSERT INTO class_probability(idDomain, IdOntologyClass, Probability)
  SELECT d.idDomain, d.IdOntologyClass, d.Documents/b.Documents Probability
  FROM (
    SELECT c.idDomain, c.IdOntologyClass, COUNT(0) Documents
    FROM class_instance a, ontology_class_members_temp c
    WHERE a.Confirmed = 1
    AND a.IdOntologyClass = c.IdConnectingOntologyClass
    GROUP BY c.idDomain, c.IdOntologyClass
    ) d, (
    SELECT f.idDomain, COUNT(0) Documents
    FROM class_instance e, ontology_class_members_temp f
    WHERE e.Confirmed = 1
    AND e.IdOntologyClass = f.IdConnectingOntologyClass
    GROUP BY f.idDomain
    ) b
  WHERE d.idDomain = b.idDomain;

  TRUNCATE TABLE class_word_probability;
  INSERT    INTO    class_word_probability(idDomain,    IdOntologyClass,    Word,
Probability)
  SELECT    a.idDomain,    a.IdOntologyClass,    a.Word,    a.Occurrence/b.AllWords
Probability
  FROM (
   SELECT b.idDomain, b.IdOntologyClass, a.Word, SUM(a.Occurrence) Occurrence
    FROM (
      SELECT a.IdOntologyClass, a.Word, SUM(a.Occurrence) Occurrence
      FROM trn_word_occurrence a
      GROUP BY a.IdOntologyClass, a.Word) a, ontology_class_members_temp b
    WHERE a.IdOntologyClass = b.IdConnectingOntologyClass
    GROUP BY b.idDomain, b.IdOntologyClass, a.Word
    ) a, (
    SELECT c.idDomain, c.IdOntologyClass, SUM(a.Occurrence) AllWords
    FROM trn_word_occurrence a, ontology_class_members_temp c
    WHERE a.IdOntologyClass = c.IdConnectingOntologyClass
    GROUP BY c.idDomain, c.IdOntologyClass
    ) b
  WHERE a.IdOntologyClass = b.IdOntologyClass
  AND a.idDomain = b.idDomain;

  DROP TABLE IF EXISTS class_word_document_probability_temp;
  CREATE TABLE class_word_document_probability_temp AS
  SELECT b.idDomain,
    b.IdOntologyClass,
    a.Word,
    CAST(COUNT(0)/c.Documents AS DECIMAL(10,9)) DocumentProbability
  FROM trn_word_occurrence AS a, ontology_class_members_temp b, (
    SELECT b.idDomain, b.IdOntologyClass, COUNT(0) Documents
    FROM class_instance a, ontology_class_members_temp b
    WHERE a.Confirmed = 1
    AND a.IdOntologyClass = b.IdConnectingOntologyClass
    GROUP BY b.idDomain, b.IdOntologyClass) c
  WHERE a.IdOntologyClass = b.IdConnectingOntologyClass
  AND b.IdOntologyClass = c.IdOntologyClass
  AND b.idDomain = c.idDomain
  GROUP BY b.idDomain, b.IdOntologyClass, a.Word;
  CREATE        INDEX        class_word_document_probability_temp_ind1        ON
class_word_document_probability_temp(IdOntologyClass);
  CREATE        INDEX        class_word_document_probability_temp_ind2        ON
class_word_document_probability_temp(Word);

  DROP TABLE IF EXISTS word_document_probability_temp;
  CREATE TABLE word_document_probability_temp AS
  SELECT a.Word,
    CAST(COUNT(0)/c.Documents AS DECIMAL(10,9)) DocumentProbability
  FROM trn_word_occurrence AS a, class_instance AS b, (
    SELECT COUNT(0) Documents
    FROM class_instance
    WHERE Confirmed = 1) c
  WHERE a.IdWebRessource = b.IdWebRessource
  AND b.Confirmed = 1
  GROUP BY a.Word;
  CREATE         INDEX         word_document_probability_temp_ind1         ON
word_document_probability_temp(Word);

  DROP TABLE IF EXISTS word_ranking_temp1;
  CREATE TABLE word_ranking_temp1 AS
  SELECT a.Word, a.IdOntologyClass, d.idDomain,

SUM(a.DocumentProbability*(LOG(a.DocumentProbability/(b.Probability*c.DocumentP
robability))))+
    SUM(1-a.DocumentProbability*(LOG(1-a.DocumentProbability/(b.Probability*(1-
c.DocumentProbability))))) Ranking
  FROM    class_word_document_probability_temp    a,    class_probability    b,
word_document_probability_temp c, ontology_class_members_temp d
  WHERE a.IdOntologyClass = b.IdOntologyClass
  AND a.Word = c.Word
  AND a.IdOntologyClass = d.idConnectingOntologyClass
  GROUP BY a.Word, a.IdOntologyClass, d.idDomain;
  CREATE INDEX word_ranking_temp1_ind1 ON word_ranking_temp1(Word, idDomain);

  DROP TABLE IF EXISTS word_ranking_temp;
  CREATE TABLE word_ranking_temp (
    IdWordRanking int(10) unsigned NOT NULL AUTO_INCREMENT,
    Word varchar(255) NOT NULL,
    IdOntologyClass int(10) unsigned NOT NULL,
    idDomain int(10) NOT NULL,
    Ranking double,
    PRIMARY KEY (IdWordRanking));

  INSERT INTO word_ranking_temp(Word, IdOntologyClass, idDomain, Ranking)
  SELECT a.Word, a.IdOntologyClass, a.idDomain, a.Ranking
  FROM word_ranking_temp1 a
  WHERE a.Ranking = (
    SELECT MAX(b.Ranking)
    FROM word_ranking_temp1 b
    WHERE b.Word = a.Word
    AND b.idDomain = a.idDomain)
  ORDER BY a.idDomain, a.Ranking DESC;

  TRUNCATE TABLE feature_selection;
  INSERT INTO feature_selection (idDomain, Word)
  SELECT a.idDomain, a.Word
  FROM word_ranking_temp a, domain b, (
    SELECT idDomain, MIN(IdWordRanking) StartPosition
    FROM word_ranking_temp
    GROUP BY idDomain) c
  WHERE a.idDomain = b.idDomain
  AND a.idDomain = c.idDomain
  AND    a.IdWordRanking    BETWEEN    c.StartPosition    AND    c.StartPosition
+FeatureNumber-1;

  DROP TABLE IF EXISTS class_word_document_probability_temp;
  DROP TABLE IF EXISTS word_document_probability_temp;
  DROP TABLE IF EXISTS word_ranking_temp1;
  DROP TABLE IF EXISTS word_ranking_temp;
  DROP TABLE IF EXISTS ontology_class_members_temp;

  UPDATE software_settings
  SET Value = (SELECT CURDATE())
  WHERE IdSoftwareSetting = 4;
END;
```