

UNIVERSITY OF SOUTHAMPTON

Semantic Web Service Generation for Text Classification

by

Stephen Wayne Ball

B.Sc. (Hons)

A thesis submitted for the degree of

Doctor of Philosophy

in the

Faculty of Engineering and Applied Science

Department of Electronics and Computer Science

January 2006

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE

DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Semantic Web Service Generation for Text Classification

by Stephen Wayne Ball

Today's Web may be considered to consist of a largely unstructured collection of documents. The Semantic Web is an effort to give information a well defined meaning so that it may be interpreted by computers. On the Web this means that metadata may be published in documents so that they may be processed in a variety of useful ways by machines. An important type of such metadata is classification information, the process of classifying documents according to some taxonomy which facilitates useful functionality on the Web such as searching and browsing.

This thesis investigates machine learning approaches to classifying text within the framework of the Semantic Web. It endeavours to advance Semantic Web research by providing a working application of Semantic Web technology thereby pinpointing limitations in the current framework and suggesting where it might be improved. This work also sets out the case for text classifiers within the Semantic Web; the requirements for this to be a working technology and, in particular, it introduces the case for Web service generation by generating classifier services on the fly using arbitrary tagged corpora.

List of Contents

Chapter 1 Introduction..... 1

1.1 Motivation..... 1

1.2 Thesis Structure 4

1.3 Contribution..... 5

1.3.1 A Semantic Web Application 6

1.3.2 Introduce Text Classifier Technology to the Semantic Web 6

1.3.3 Novel Approach..... 6

1.4 Declaration..... 6

Chapter 2 The Semantic Web and Its Services 7

2.1 Introduction..... 7

2.2 The Context for Semantic Web Services..... 7

2.2.1 The Semantic Web..... 7

2.2.2 XML Standards..... 8

2.3 The Current State of the Semantic Web 10

2.3.1 Some Standards for a Semantic Web..... 10

2.3.2 A Brief Portrait of the Semantic Web..... 13

2.4 The Current State of Web Service Standards 15

2.5 The Current State of Semantic Web Services..... 18

2.5.1Introduction..... 18

2.5.2 OWL-S..... 18

2.5.3 The Web Service Modeling Framework (WSMF) 20

2.5.4 IRS-II 21

2.6 Summary..... 23

Chapter 3 Machine Learning and Text Classification..... 24

3.1 Introduction..... 24

3.2 Text Classifiers 24

3.3 Text Preparation..... 25

3.4 Feature Selection..... 26

3.5 Naïve Bayes 27

3.5.1 Multinomial Naïve Bayes	28
3.5.2 Multivariate Naïve Bayes	30
3.6 Support Vector Machines	30
3.6.1 Introduction.....	30
3.6.2 The Simplest Support Vector Machine.....	31
3.6.3 Allowing for Imperfect Data.....	34
3.6.4 Kernels	35
3.6.5 Solving the Dual Problem.....	36
3.6.6 Multi-Class SVMs	37
3.7 Summary.....	38
Chapter 4 Classifier Performance.....	39
4.1 Introduction.....	39
4.2 A Software Framework.....	39
4.2.1 Background.....	39
4.2.2 A Software Framework for Classifiers	41
4.3 Performance Measures.....	43
4.4 Reuters-21578	44
4.5 The ACM Corpus.....	50
4.6 Discussion of Results.....	57
4.7 Summary.....	58
Chapter 5 ClassAKT: A Semantic Web Service	60
5.1 Introduction.....	60
5.2 Classifier Construction	60
5.3 A Semantic Web Service	63
5.4 Summary.....	64
Chapter 6 Document Archives	66
6.1 Introduction.....	66
6.2 The Development of Digital Archives.....	66
6.3 Open Archival Information System (OAIS).....	68
6.4 Open Archive Initiative (OAI).....	72
6.5 The Dublin Core Metadata Initiative (DCMI).....	74

6.6 Summary..... 75

Chapter 7 Ontologies..... 78

7.1 Introduction..... 78

7.2 An Ontology for a Digital Archive 79

7.3 Extending the Digital Archive Ontology 83

7.4 An Ontology for Text Classification 85

7.4.1 Design of a Document 88

7.4.2 Design of a Taxonomy for Processing Resources 90

7.4.3 Pulling Everything Together..... 91

7.5 Summary..... 94

Chapter 8 Web Service Generation 96

8.1 Introduction..... 96

8.2 Web Service Generation 96

8.2.1 Semantic Web Developers..... 100

8.2.2 Researchers 100

8.3 A Web Service Generator Implementation..... 101

8.3.1 Overview..... 101

8.3.2 FeatureCreationService..... 102

8.3.3 ClassifiersServiceEx 105

8.4 Outstanding Issues 106

8.4.1 Service Advertisement..... 106

8.4.2 Describing Services in an Ontology 108

8.5 Ontology Mapping..... 109

8.6 Deployment Issues 116

8.7 Summary..... 118

Chapter 9 Conclusions and Future Work 119

9.1 Conclusions..... 119

9.1.1 Introduce Text Classifier Technology to the Semantic Web..... 119

9.1.2 A Semantic Web Application 120

9.1.3 Novel Approach..... 121

9.2 Future Work..... 122

Appendix A	ClassAKT Service Ontologies	124
A.1	Guidance Notes	124
A.2	Service Ontology	124
A.3	Service Profile.....	125
A.4	Service Process	127
A.5	Service Grounding	129
A.6	Service WSDL Grounding.....	131
Appendix B	An Ontology for a Digital Archive	133
B.1	Guidance Notes	133
B.2	The OWL Ontology	133
Appendix C	An Ontology for the ACM Portal	136
C.1	Guidance Notes	136
C.2	The OWL Ontology	136
Appendix D	An Ontology for Text Classification	139
D.1	Guidance Notes.....	139
D.2	The OWL Ontology	139
Appendix E	An Ontology for Classifier Service Generation.....	144
E.1	Guidance Notes	144
E.2	ClassifierCreation_CreateSession	145
E.3	ClassifierCreation_AddClass	147
E.4	ClassifierCreation_AddFeaturesDocument.....	150
E.5	ClassifierCreation_AddInstance.....	153
E.6	ClassifierCreation_CreateClassifier	156
E.7	Classifier_ClassifyTemplate	159
Glossary		163
Machine Learning		163
Semantic Web		164
Bibliography		166

List of Figures

Figure 2.1: Some example XML	8
Figure 2.2: XML Schema for XML in Figure 2.1	9
Figure 2.3: Semantic Web Layer Cake (after original presentation by Tim Berners-Lee at XML2000 Conference).....	10
Figure 2.4: Some RDF/XML Describing Hugh Glaser	11
Figure 2.5: An RDF Graph Describing Hugh Glaser	11
Figure 2.6: RDF Schema for the RDF of Figure 2.4	12
Figure 2.7: OWL-S Service Upper Ontology	18
Figure 2.8: UPML Framework	22
Figure 3.1: Two Classes Separated by a Plane with Maximal Margin.....	32
Figure 4.1 Class Hierarchy for Classifiers Architecture.....	41
Figure 4.2 Multinomial Naïve Bayes Classifier Performance Varying Files for Reuters-21578 Classes	47
Figure 4.3 Multinomial Naïve Bayes Classifier Performance Varying Features for Reuters-21578 Classes.....	47
Figure 4.4 Multivariate Naïve Bayes Classifier Performance Varying Files for Reuters-21578 Classes	48
Figure 4.5 Multivariate Naïve Bayes Classifier Performance Varying Features for Reuters-21578 Classes.....	48
Figure 4.6 SVM Classifier Performance Varying Files for Reuters-21578 Classes	49
Figure 4.7 SVM Performance Varying Features for Reuters-21578 Classes.....	49
Figure 4.8: Extract from the ACM Computer Classification System.....	50
Figure 4.9 Multinomial Naïve Bayes Classifier Performance Varying Files for ACM ‘C’ Classes	53
Figure 4.10 Multinomial Naïve Bayes Classifier Performance Varying Features for ACM ‘C’ Classes.....	53
Figure 4.11 Multivariate Naïve Bayes Classifier Performance	

Varying Files for ACM ‘C’ Classes 54

Figure 4.12 Multivariate Naïve Bayes Classifier Performance

Varying Features for ACM ‘C’ Classes..... 54

Figure 4.13 SVM Classifier Performance

Varying Files for ACM ‘C’ Classes 55

Figure 4.14 SVM Clasifier Performance

Varying Features for ACM ‘C’ Classes..... 55

Figure 4.15 Multinomial Naïve Bayes Classifier Performance

Varying Files for ACM ‘A’ to ‘K’ Classes..... 56

Figure 4.16 Multinomial Naïve Bayes Classifier Performance

Varying Features for ACM ‘A’ to ‘K’ Classes..... 56

Figure 6.1: Generation of an Information Object 69

Figure 6.2: OAIS Functional Entities 70

Figure 6.3: Information Package Concepts and Relationships..... 71

Figure 6.4: Some metadata in Dublin Core formatted within RDF/XML..... 75

Figure 7.1 The ABC Ontology as a Core Ontology 80

Figure 7.2 The ABC Class Hierarchy..... 80

Figure 7.3 An Ontology for a Digital Archive..... 82

Figure 7.4 Mapping Documents to Classifications..... 84

Figure 7.5 Key Inputs and Outputs of a Feature Extractor..... 87

Figure 7.6 Key Inputs and Outputs of a Classifier Generator 87

Figure 7.7 Document Representation in Text Classification Ontology..... 88

Figure 7.8 Processing Resource Representation in Text Classification Ontology 90

Figure 7.9 Document Types during Classifier Generation..... 93

Figure 8.1 Semantic Web Classifier Service Generator 99

Figure 8.2 Schematic Composite Process for Feature Generation 103

Figure 8.3 Schematic Composite Process for Classifier Service Generation..... 104

Figure 8.4 OWL-S Service Advertisement for a Classifier Service 107

Figure 8.5 Top level System Classes in a Notional OWL Framework..... 109

Figure 8.6 MAFRA Conceptual Architecture

(adapted from [Maedche et al., 2002]) 111

Figure 8.7 MAFRA Mapping of Two Ontologies
(adapted from [Maedche et al., 2002]) 112

Figure 8.8 MAFRA Mapping of Text Classification and Digital Archive Ontologies .. 114

List of Tables

Table 4.1: Contingency Table..... 43

Table 4.2 Some Results for ‘A’ to ‘K’ Classes of the ACM Corpus using a SVM..... 52

Table 5.1: Class Distribution of ACM Documents..... 62

Table 6.1: Simple Dublin Core Metadata Element Set, Version 1.1 74

Acknowledgements

Much thanks are due to my supervisors Hugh Glaser and Nick Gibbins for their support of my work and thoughtful ideas. Thanks also to Craig Saunders for his guidance in many technical aspects relating to classifier technology.

Thanks also to the Engineering and Physical Sciences Research Council (EPSRC) whose financial assistance has made this thesis possible.

Chapter 1 Introduction

1.1 Motivation

The Web of today can be regarded as a largely unstructured collection of HTML pages. Where there is organisation, for example an interface which uniformly presents the contents of a database, resultant resources do not share standards and remain effectively as data islands unconnected with each other. The Semantic Web initiative [Berners-Lee et al., 2001] is an effort to address these shortcomings, to employ the current best practice in areas as diverse as knowledge technologies and Web services, so that standards and technologies are available to make ‘knowledge resources’ on the Web. The result is that information on the Web becomes machine-processable and a variety of knowledge-driven applications may be developed which facilitate business processes and the user experience generally.

Knowledge resources can be constructed from data originating from various sources, for example, from traditional databases or Web harvesting (using various tools to mine data from the existing Web). In addition, some knowledge may be assembled manually. The problem is that, in general, manually generating knowledge becomes prohibitively time consuming and expensive when the data set becomes large. Techniques of generating this knowledge automatically then become valuable. One example, and the basis of a service framework introduced in this thesis, is the use of text classification technologies to make use of an established classification system and a

body of existing documents which have previously been classified, to build classifiers which enable further classification to be automated.

Text classification techniques have been studied for decades. More recently interest in the field has greatly increased due to the availability of powerful computer hardware and greater awareness of possible applications of the technology. In particular, a machine learning approach has proved to be effective at producing classification systems with accuracy comparable to that achieved via manual classification. In machine learning [Mitchell, 1997], a general inductive process is employed to build the classifiers. Many algorithms are available from diverse areas of Computer Science, and in this work two techniques have been focused on. Naïve Bayes, a probabilistic classifier [Lewis, 1998], has been adopted because of its history of producing good results and general popularity as a ‘baseline’ classifier. Support Vector Machines (SVMs) [Joachims, 1998], originating from statistical learning theory, are reported to produce some of the highest quality results known in text classification and may be considered state of the art.

The Advanced Knowledge Technologies project (AKT¹), funded by the Engineering and Physical Sciences Research Council (EPSRC), is scheduled to last for 6 years and is a collaboration between research groups at the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University. AKT is an endeavour that directly requires the adoption of current best practice in the area of Semantic Web research and considers there are six challenges to address in order to ease fundamental bottlenecks in the engineering and management of knowledge:

- **Acquiring Knowledge:** the challenge of getting hold of the information that is around and turning it into knowledge by making it usable.
- **Modeling Knowledge:** straightforward placeholders for the acquired knowledge coming in and to represent the knowledge so that it can be used for problem-solving. The use of ontologies is particularly important in the area of modeling knowledge.
- **Reusing Knowledge:** Understanding how to find patterns in knowledge, to allow for its storage in a library so that it can be reused when circumstances permit.

¹ See www.aktors.org

- **Retrieving Knowledge:** this challenge addresses the issue of recovering a subset of content from a (typically) large knowledge store that is relevant to a particular problem or task.
- **Publishing Knowledge:** delivering knowledge in the right form, at the right place, to the right person at the right time.
- **Maintaining Knowledge:** keep the knowledge repository useful by maintaining it over time.

In particular, the AKT Reference Ontology¹ has been developed which models UK Computer Science departments. An ontology is a specification of the generic concepts, attributes, relations and axioms of a domain and, in connection with this work, is represented in terms of the Web Ontology Language (OWL), a language grounded in a description logic. A knowledge base has been created and populated with OWL triples (triples are statements with subject, verb and object) described in terms of the reference ontology. The actual knowledge base is implemented with Triplestore² a knowledge base infrastructure which contains the triples and which may be queried against the knowledge it stores.

The AKT Reference Ontology enables the inclusion of metadata about papers published in scientific journals into the knowledge base. It incorporates the ACM Computing Classification System³ to allow a classification to be associated with these papers. The ACM classification system contains top level classes such as Hardware and Computer Systems Organization and is hierarchical so that each of these top level classes contains sub-classes in a tree-like structure. For the papers published by the ACM, a manual classification is available via the ACM portal. However, no such classification exists for the vast majority of papers known to Triplestore. This is a significant shortcoming. Comprehensive classification metadata relating to document resources in a knowledge base enables an important set of features such as searching and browsing. An objective of the work undertaken in this thesis is to rectify this situation. By making use of papers and their existing classification maintained in the ACM portal, classifiers will be investigated and built which will enable the appropriate classification

¹ See www.aktors.org/publications/ontology/

² See triplestore.aktors.org

³ See portal.acm.org

fields to be populated to a high standard for papers within the Triplestore knowledge base.

The AKT Reference Ontology models UK Computer Science departments and its Triplestore functions effectively as a stand alone knowledge repository. It takes no account of integrating a service architecture into its framework nor does it consider the related issue of integration of semantically different domains. In particular, introducing text classification services requires the interoperation of an ontology such as the AKT Reference Ontology with an ontology which deals specifically with the domain of machine learning based text classifiers. Little investigation has been made to date to investigate such issues within the context of a working system in the fledgling Semantic Web. The work here has the opportunity to see how these issues may be overcome, to pull together many of the ideas which are currently being researched and put into the public domain an implementation which provides insights into making the Semantic Web work in practice.

1.2 Thesis Structure

This thesis begins by introducing the key areas of technology, namely the Semantic Web and Machine Learning based text classification. Then, from the perspective of these two technologies, it expands upon and investigates issues relating to making text classifiers a viable application on the Semantic Web.

Chapter 2, The Semantic Web and Its Services, outlines the current state of the Semantic Web and key technologies that have been introduced for it. In particular, current Semantic Web service architectures are described as such an architecture is required for much of the work to follow.

Chapter 3, Machine Learning and Text Classification, introduces some background theory behind machine learning based text classifiers. Focus is placed on a study of two key technologies, naïve Bayes and Support Vector Machines.

Chapter 4, Classifier Performance, introduces a software framework which at this stage of development is useful for testing classifiers. Test results are then published for two corpora, the ACM corpus and Reuters-21578. This testing is made from the perspective of producing results which may be seen as providing guidelines for producing classifiers for new corpora.

Chapter 5, ClassAKT: A Semantic Web Service, describes a Semantic Web service which classifies unseen documents according to the top two levels of the ACM classification tree.

Chapter 6, Document Archives, explores some of the key standards in place to model and deploy digital archives. The domain of digital archives is considered a key area in which text classifiers will be required to operate.

Chapter 7, Ontologies, researches best practice to introduce two key ontologies required in this work in order to root text classifiers in the Semantic Web. These are an ontology for digital archives and an ontology for classifier services.

Chapter 8, Web Service Generation, pulls together the work of the previous chapters by describing a novel service which can take an arbitrary corpus and classification scheme and automatically generate a classifier Semantic Web service from this specification. Related issues are explored, in particular enabling the interoperability of the classifier Semantic Web services with the ontology for digital archives.

Chapter 9, Conclusions and Future Work, concludes this thesis by drawing out a portrait of what has been achieved and exploring how this work may be continued to provide a useful, viable application on the Semantic Web.

1.3 Contribution

This thesis provides three contributions.

1.3.1 A Semantic Web Application

Research into the Semantic Web has focused on the fundamental technologies required to make it work: for example RDF, OWL and an initial effort towards a service framework in the form of OWL-S. Semantic Web infrastructure is far from being complete and consequently any work which seeks to place real Semantic Web applications on the Web will be a valuable and worthwhile effort. It is the intention of this thesis to contribute to this important requirement, to document what may be achieved with current technology and pinpoint key areas which must be addressed if the Semantic Web is to work.

1.3.2 Introduce Text Classifier Technology to the Semantic Web

The Web is a largely unstructured collection of textual documents whereas the provision of metadata to Web content is a key application of Semantic Web technology. It follows that a valuable and important application of the Semantic Web is the provision of classification metadata which will enable functionality on the Web such as searching and browsing. This work will address the issues surrounding the automatic generation of such classification metadata making use of a machine learning approach to automatically generate such metadata and thus placing this important topic and its solution in the public domain.

1.3.3 Novel Approach

Arising as an implementation requirement out of the first two contributions, it is found that dynamic creation of services which enable the generation of classifiers based on arbitrary tagged corpora is a fundamental and essential requirement. This novel ‘Web service generation’ is introduced as an important design pattern which is likely to arise in various domains in service environments in the emergent Semantic Web.

1.4 Declaration

This thesis is based upon work undertaken by the author. What is presented is the original work of the author and where previous work has been of influence, this has been duly recognised and referenced. This work has been supported by EPSRC studentship grant number GR/POO437/01.

Chapter 2 The Semantic Web and Its Services

2.1 Introduction

The effort towards a Semantic Web is still relatively new and has certainly not yet achieved a sufficient level of maturity such that it may be applied effectively and easily on today's Web. However, numerous initiatives are underway and the function of this chapter is to portray an outline of what has been achieved to date. Of key importance is the representation of knowledge on the Web and of particular interest in this work is the current state of Semantic Web service architectures that enable functional entities, such as text classifiers, to be invoked on the Web. In addition, the underlying importance of and connection with existing Web standards will be addressed.

2.2 The Context for Semantic Web Services

2.2.1 *The Semantic Web*

The Semantic Web initiative [Berners-Lee et al., 2001] may be considered to be an effort towards evolving today's Web (which may be considered a collection of mostly HTML pages designed for human consumption) into one which contains more structure and meaningful data (or *knowledge*) and which in particular, enables the interpretation of this knowledge by computers.

Certainly, one can envisage in such a scenario, the usefulness of *agents*: autonomous software components which are able to interpret this knowledge to perform useful tasks. For example a person may have a personal agent which is capable of

coordinating that individual's diary. The agent may be tasked with arranging travel arrangements for a holiday, in which case it may communicate with various *Semantic Web services* [McIlraith et al., 2001], software objects which can perform some specific task, to arrange flights and hotels. Clearly the requirement for agents and services, objects which can interpret and manipulate the published knowledge, will be crucial to the development of a successful Semantic Web.

2.2.2 XML Standards

Knowledge representation is considered later in this chapter. Fundamentally to describe knowledge in the context of the work undertaken in this thesis, knowledge representation builds on XML notation [W3C, 2004a], a text format derived from SGML [ISO, 1986]. Some example XML is shown in Figure 2.1. This XML describes a possible set of classifications (A, A.0, etc) for some document. Metadata (classes, class) which describe the meaning of the actual data is formatted between angle brackets.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<classes xmlns="http://classakt.actors.org">
  <class>http://www.acm.org/class/1998/A</class>
  <class>http://www.acm.org/class/1998/A.0</class>
  <class>http://www.acm.org/class/1998/H</class>
  <class>http://www.acm.org/class/1998/H.3</class>
</classes>
```

Figure 2.1: Some example XML

Underpinning XML are a number of standards which are important for the consideration of the Semantic Web:

- *XML Namespaces* [W3C, 2004b]: Makes use of an Internationalized Resource Identifier (IRI, similar to a URI) to distinguish XML schemes when a number of them occur in the same document. In Figure 2.1, the xml is defined within the default namespace "http://classakt.actors.org". Now for example adding a new 'class' element belonging to a new XML scheme could be distinguished by its own namespace IRI.

- *XML Schema* [W3C, 2001a]: Written in XML, an XML Schema provides a means for defining the structure and content for XML documents. An example XML schema for the XML in Figure 2.1 is given in Figure 2.2.

```
<xsd:schema targetNamespace="http://classakt.aktors.org/services/owl-
s/0.9/ClassAKT.xsd">
  <xsd:element name="classes" type="ClassesType"/>
  <xsd:complexType name="ClassesType">
    <xsd:sequence>
      <xsd:element name="class" type="xsd:anyURI"
minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Figure 2.2: XML Schema for XML in Figure 2.1

- *XPath* [W3C, 1999a]: A language for addressing parts of an XML document. It is similar to file system addressing, so that for example in Figure 2.1, ‘/classes/class[1]’ will pick out the first ‘class’ node.
- *XPointer* [W3C, 2001b]: Builds on XPath, XPointer is an addressing scheme for individual parts of an XML document. A URI typically points to a whole document whereas an XPointer allows selection of specific nodes within an XML document.
- *XLink* [W3C, 2000a]: Provides a mechanism to connect elements in the same or different XML document. The functionality is an extension to that offered by the anchor in HTML.
- *XSLT* [W3C, 1999b]: The Extensible Stylesheet Language Transformations provides a tree orientated transformation language for transmuting instances of XML using one vocabulary into plain text, HTML or XML using a different vocabulary.

2.3 The Current State of the Semantic Web

2.3.1 Some Standards for a Semantic Web

As mentioned earlier, the Semantic Web may be considered an initiative to place more structured and meaningful data on the Web, data which may be interpreted by computers in an intelligent way. HTML¹, the pervasive standard for Web publishing, is an SGML application that was designed essentially for enabling the presentation of documents on the Web. Its ability to describe metadata about Web resources is very limited such that it is unsuited to the task of publishing *knowledge*. Consequently, new formats and standards are required. Figure 2.3 enumerates the key components envisaged for the new framework which it is intended will enable the implementation of a Semantic Web.

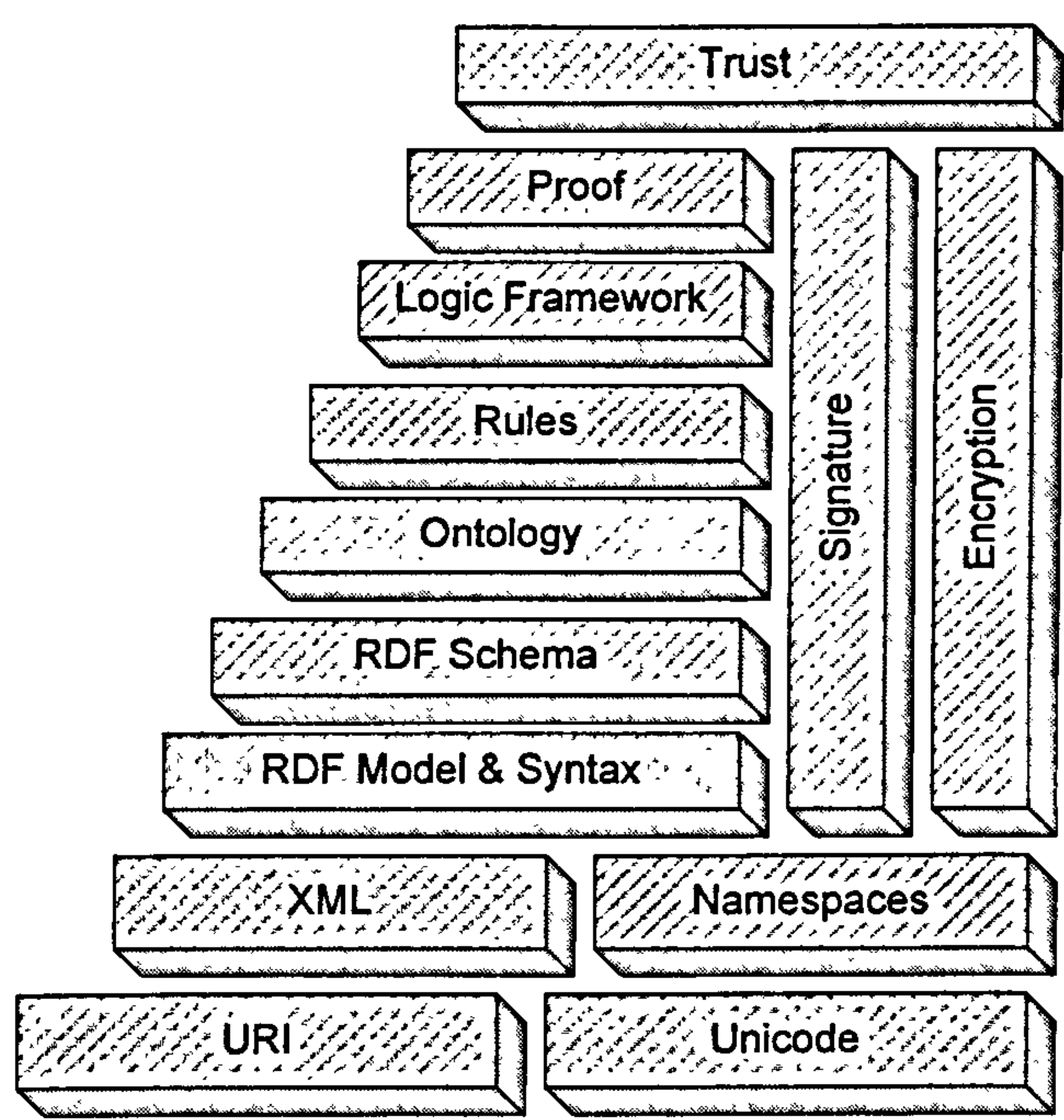


Figure 2.3: Semantic Web Layer Cake

(after original presentation by Tim Berners-Lee at XML2000 Conference)

¹ See www.w3.org/MarkUp/

RDF is a conceptually simple XML based language designed to describe resources on the Web with a particular aim of enabling those resources to be processed by applications. It also may be shown in graphical form with nodes and arcs representing the resources and their properties and values. Some sample RDF is shown in Figure 2.4 and the corresponding graphical view in Figure 2.5. A good general introduction may be found in the W3C's Primer [W3C, 2004c]. Essentially, RDF models knowledge as a triple consisting of a subject, predicate and object and makes extensive use of URIs to identify particular resources. So, for example in Figure 2.4, the subject would be the person identified by the URI <http://www.ecs.soton.ac.uk/info/#person-00021> and there are two properties, for example the first has a predicate of 'ns0:full-name' and an object, which happens to be a string literal, of 'Hugh Glaser'.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ns0="http://www.aktors.org/ontology/portal#">
<rdf:Description rdf:about="http://www.ecs.soton.ac.uk/info/#person-
00021">
  <ns0:full-name>Hugh Glaser</ns0:full-name>
  <ns0:has-email-address>hg@ecs.soton.ac.uk</ns0:has-email-address>
</rdf:Description>
</rdf:RDF>
```

Figure 2.4: Some RDF/XML Describing Hugh Glaser

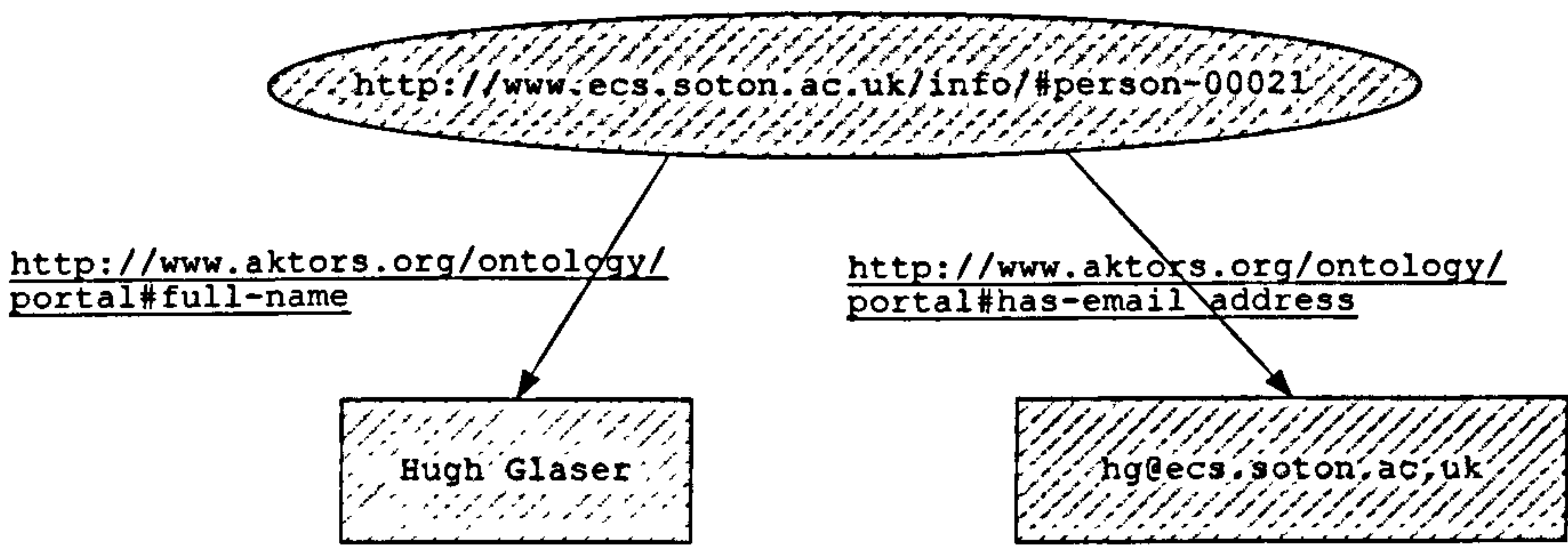


Figure 2.5: An RDF Graph Describing Hugh Glaser

RDF Schema (RDFS) [W3C, 2004d] is built on top of RDF and indeed is described in terms of it. It can be considered by analogy with a modern object oriented programming language such as Java: RDF Schema enables the definition of *classes*, permits sub-classing and provides the means of specifying and typing *properties*. Figure 2.6 is the RDF Schema for the RDF of Figure 2.4. Indeed, RDF Schema may be considered as a primitive ontology/knowledge representation language. An *ontology* may be considered as an explicit specification of a conceptualization and as RDFS shows, is a means of modelling a specific domain.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<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://www.aktors.org/ontology/portal#">

  <rdfs:Class rdf:ID="Person"/>

  <rdf:Property rdf:ID="full-name">
    <rdfs:domain rdf:resource="#Person" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </rdf:Property>

  <rdf:Property rdf:ID="has-email-address">
    <rdfs:domain rdf:resource="#Person" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </rdf:Property>
</rdf:RDF>
```

Figure 2.6: RDF Schema for the RDF of Figure 2.4

The Semantic Web community has appreciated the limited capability of RDFS and has sought to build on this technology to provide greater expressivity while understanding the requirement of providing a language which is amenable to *automated reasoning*, enabling machines to infer knowledge or, for example, in the case of Semantic Web Services to enable automated Web service composition. With these considerations

in mind, researchers have investigated exploiting existing work on *description logics* to provide a richer Web ontology language. Description logics [Calvanese et al., 2002, Baader et al., 2002] form a family of both class-based and logic-based knowledge representation languages, that permit modelling a particular domain in terms of classes, objects and relationships between them and which also permit reasoning about them. The current language is the OWL Web Ontology Language which builds on RDF(S) and which has itself been derived from an earlier effort, DAML+OIL [Horrocks, 2002]. OWL [W3C, 2004e] provides a number of additional primitives which facilitate the creation of richer, more fully featured ontologies. Examples of these primitives would be class constructors such as ‘intersectionof’ and ‘unionof’ and axioms such as ‘equivalentClass’, ‘inverseOf’ and ‘TransitiveProperty’.

OWL is itself divided into three sublanguages, namely OWL Full, OWL DL and OWL Lite. OWL Full is in fact the entire OWL language and provides free, unconstrained use of RDF constructs. However, OWL Full offers no computational guarantees and it is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full. Although OWL Full and OWL DL support the same set of OWL language constructs, their difference lies in restrictions placed in OWL DL on the use of some of those features and on the use of RDF features. For example, in OWL DL while a class may be a subclass of many classes, a class cannot be an instance of another class. However, OWL DL offers computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL Lite has a lower formal complexity than OWL DL and is designed primarily for those users implementing a classification hierarchy with simple constraints.

2.3.2 A Brief Portrait of the Semantic Web

With standards in place, it is interesting to consider what the Semantic Web will look like. It may be considered that metadata will be published in two key and closely interrelated ways, namely as markup distributed around the Web and knowledge available via knowledge bases (see for example [Heflin & Hendler, 2001]).

Publishing metadata on the Web facilitates easy interpretation via a Web browser. At any given Web site, metadata may be published pertaining to a number of different

topic domains. That is metadata exists pertaining to a number of different ontologies. Consider a person's Web site where there may be metadata corresponding to that person's interests. Each interest is likely to be described by a separate ontology, yet there is a benefit to collecting all of that person's metadata at one location. Indeed, perhaps individuals may in the future be represented on the Web by agents at their personal Web site which manage knowledge about the individual providing various productivity tools to the site owner (as a small example, managing information about finance or subscriptions) as well as publishing certain, unprivileged information available to the Web community in general. With the introduction of appropriate browsing tools, the casual human browser can navigate this metadata to gain an idea of what interests the author. Links to resources in the metadata can point to useful sites containing related metadata.

An initiative in this area is the Friend of a Friend (FOAF) project (<http://www.foaf-project.org/>). This is an RDF/XML Semantic Web vocabulary typically used for publishing personal information, name, address, employment, interests, etc. Users publish their personal metadata on their own site in one or more FOAF files which can then be read for use within a variety of applications, for example, to create community directories. In particular, while creation of the metadata is managed in a distributed fashion (under the control of the author), it may be aggregated into a knowledge base where it may be explored and cross linked.

Further, an important issue in the domain of distributed metadata is that of *ontology mapping* [Kalfoglou & Schorlemmer, 2003, Maedche et al., 2002]. Ontology mapping refers to various techniques which enable the linking of concepts and properties across different ontologies. For example, the 'person' concept is likely to be represented in all ontologies representing personal details (and each ontology is likely to have a different representation of personal details). Yet, for a computer to be able to make the link between person concepts in different ontologies, such a relationship must be explicitly expressed. Mapping ontologies in this way enables many useful features of knowledge based application by linking various topic domains together. For example, navigation across metadata is facilitated and it is possible to aggregate knowledge together relating to a particular concept. As a further trivial example, an author of

metadata may type in their personal details initially, but where such information is required in a new ontological domain, the mapping process can infer that the personal details are the same in the new area and fill necessary fields in automatically.

On the other hand, knowledge may be stored centrally in *knowledge bases*. Such a system, 3Store [Harris & Gibbins, 2003], has been developed within the AKT research project (<http://www.aktors.org/akt/>) at Southampton University to hold knowledge pertaining to UK Computer Science departments. Such systems need to be scalable and robust to manage large quantities of RDF (or OWL) statements. Centralising the knowledge is the most appropriate technique to enable querying and *inference* across the knowledge for various clients such as computer agents or human users.

Knowledge available in some arbitrary knowledge base is likely to be represented somewhere on the Web as published metadata. Indeed there are pros and cons to maintaining metadata in either a centralized or distributed fashion. When centralized, the knowledge of the subject domain may be accessed and queried in its entirety. When distributed across Web sites metadata may be browsed within the context of its owner (for example, conventional Web site) and metadata pertaining to various ontologies in which the site owner has a stake may be considered together.

How the distribution of metadata will in reality look across centralized and distributed stores is a question which will perhaps be answered only as the Semantic Web takes shape. At this stage, it is perhaps wise to consider that the whole breadth of applications is likely to occur, from fully centralized to fully distributed collections of metadata pertaining to specific ontologies. However, mainstream Semantic Web solutions will perhaps replicate and link their knowledge in both knowledge bases and across the Web.

2.4 The Current State of Web Service Standards

Efforts to solve issues surrounding the effective use of Web services and to produce standards for such services, have been in operation since well before their importance was stated within the context of the forthcoming Semantic Web. If a particular Web service standard is not directly adopted for Semantic Web services, then it is likely at least to exert a strong influence on the shape of standards which will emerge. It is

therefore important to consider such efforts and standards here. Standards may be considered within the context of their role and these roles are enumerated below.

Communication. A standard is required for the transfer of messages across the Web. The current ubiquitous standard is the Simple Object Access Protocol (SOAP) [W3C, 2000b]. SOAP is an XML based message layout specification which defines a binding to HTTP as the underlying communication protocol. It is a lightweight protocol designed to be platform independent and Internet friendly (enabling, for example, messages to pass through firewalls). In addition, three object server technologies may be considered to be predominant: CORBA, EJB and COM+ and these respectively use Internet Inter-ORB Protocol (IIOP), Remote Method Invocation (RMI) and Distributed Component Object Model (DCOM) as their primary method to pass information to and from objects. SOAP is not simply another alternative but can actually be the most convenient bridge to enable interoperability between the various protocols.

Grounding. A grounding protocol provides a computing process with sufficient information for automatically invoking some Web service. The Web Service Description Language (WSDL) [W3C, 2001c] is the most prevalent standard used in this area. It describes a service, in XML, as an abstract set of network endpoints (ports). The concrete network protocol and data format specification for a particular port type (typically defined with XML Schema) constitute a binding, and a set of ports constitutes a service.

Advertisement. A service provider, typically a business, needs to make his Web service available to interested clients just as clients need to be able to find a particular service on the Web. Universal Description, Discovery and Integration (UDDI) [UDDI.org, 2000] is the prevalent standard which fulfils these requirements. It provides a model for registries, where service information is advertised, plus a set of APIs which define how the registries may be accessed. A given service will be published to a registry with information about the service provider, for example business details (“white pages”), details of any taxonomies which aid in searching the registry for, e.g. a particular industry

or product category (“yellow pages”) and technical descriptions to help to connect to the service (“green pages”).

Process. Consider selecting and purchasing books from a Web retailer such as Amazon.com. In general, a customer will select one or more books, add them to the shopping cart and then proceed to the checkout to make a purchase. This procedure may be modelled by two simple Web services, one to add a book to the cart and a second to action the purchase. In general, such a procedure may be required to be modelled as a single composite service, that is, *Web service composition* of the simpler services is required. In this example, a composite service is assembled which says a particular service is run in a loop and on completion a second service is invoked to complete the purchase. Such a composite service allows a service provider to publish services at his choice of abstraction and requires a rich language of primitives to describe workflow patterns (for example, looping and concurrency) as well as communication patterns (to include various synchronous and asynchronous patterns).

Unlike other standards presented in this section, a number of standards have arisen to describe process and it can be argued that not only have these failed to reach a maturity but there has been a lack of critical examination of their completeness and effectiveness [van der Aalst, 2003]. Standards representing languages which attempt to make a contribution to this field include BPML (and the earlier WSCI), BPSS (part of ebXML), WS Choreography and BPEL4WS. Of these perhaps BPEL4WS [IBM, 2003] is currently receiving most attention. It is an initiative derived from the block-structured language XLANG from Microsoft and the directed graph architecture of WSFL from IBM. Although BPEL4WS is a rich and descriptive language, as a recent study has shown [Wohed et al., 2002], there are a number of workflow and communication patterns which it does not support reflecting that work in this domain is far from complete.

2.5 The Current State of Semantic Web Services

2.5.1 Introduction

This section surveys the chief initiatives currently being undertaken to establish a service infrastructure for the Semantic Web, namely OWL-S, WSMF and IRS-II. A further introduction and comparison of these methods may be found in [Cabral et al., 2004].

2.5.2 OWL-S

The definition of a framework for Semantic Web services must address the same issues faced by the more conventional Web service framework described in section 4.3: that is communication, grounding, advertisement and process. However, to be practical it additionally is required to conform to existing Semantic Web standards and to integrate seamlessly with them. These are the requirements and objectives of the OWL-S (formerly DAML-S) initiative presented by the OWL Services Coalition, a collection of research and industrial workers from around the world. OWL-S [OWL-S, 2004, Martin et al., 2004] is defined by a set of ontologies described in OWL currently at version 1.1 which are published at [OWL-S, 2005]. Semantic Web services are described by publishing their description in terms of OWL-S. Figure 2.7 is an illustration of the relationship between these component ontologies.

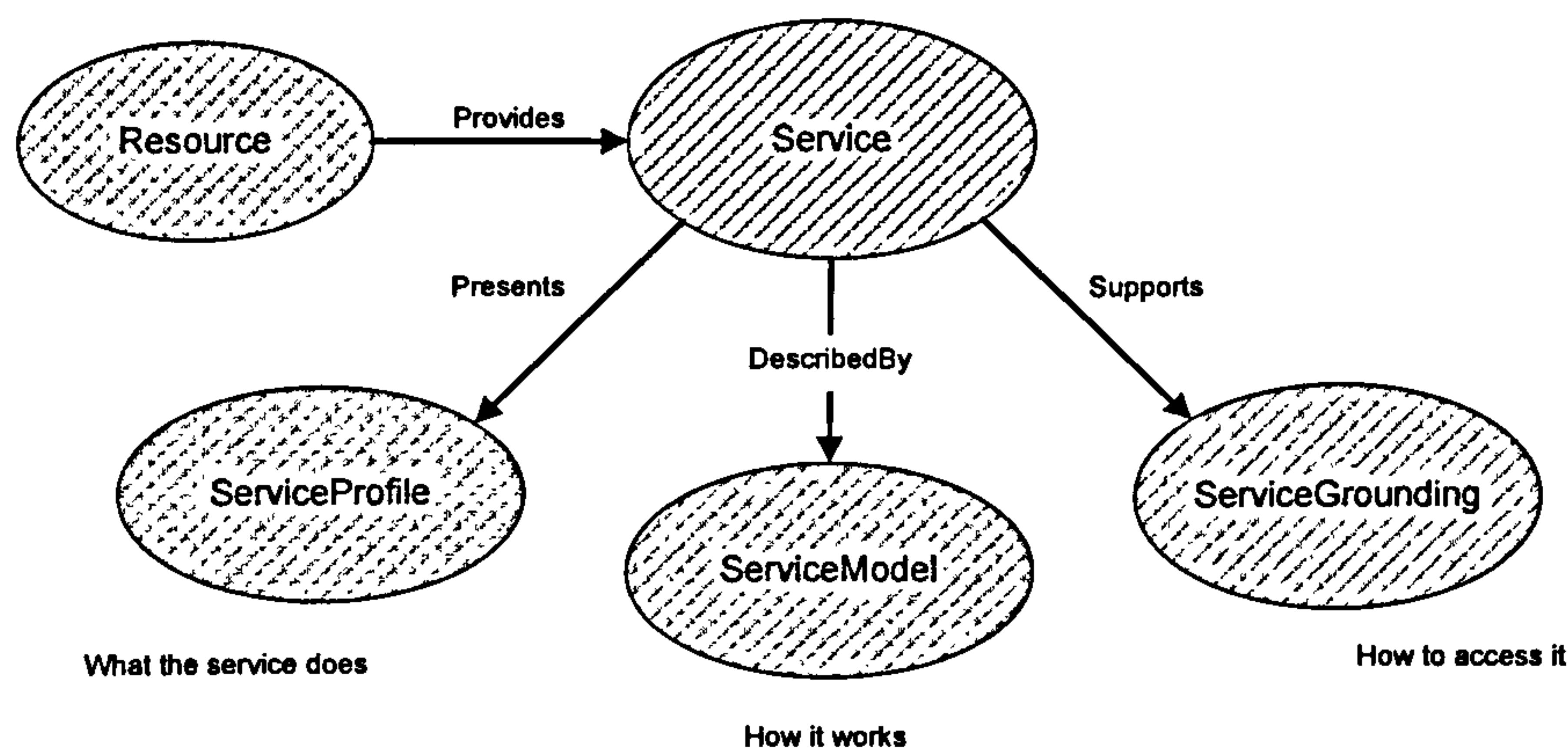


Figure 2.7: OWL-S Service Upper Ontology

The function of each of these component ontologies is as follows:

Service. Effectively a hub which contains OWL statements pointing to the other three ontologies.

ServiceProfile. The service advertisement [Paolucci et al., 2002a], provides similar information to that made available with UDDI. Indeed work has been carried out which shows how the OWL-S profile may be mapped to a UDDI service description [Paolucci et al., 2002b]. An OWL-S service description may provide more than one service profile, each one advertising the service to a varying level of abstraction.

ServiceModel. OWL-S defines its own process model with the objective that sufficient information is provided such that, for a given Semantic Web service, some automated client on the Semantic Web can interpret the ServiceModel and invoke the service without human intervention. The ServiceModel has the idea of *atomic* and *composite* services. An atomic service obeys simple request/response behaviour whereas a composite process may have a more complicated semantics along the lines of a BPEL4WS service description. However, the OWL-S ServiceModel does not contain as rich a model as BPEL4WS. For example, the ClassAKT web service for automated text classification according to the ACM hierarchy has an OWL-S service description published at <http://classakt.aktors.org/classifier> together with the service itself. There are currently a number of features missing from the OWL-S ServiceModel which are required by ClassAKT to make it more fully described, for example asynchronous communication (and an associated model of session variables) and exception handling. For a further example, see [Sabou et al., 2003] where a number of limitations of the current model were reported during the practical implementation of a number of Semantic Web services in DAML-S.

ServiceGrounding: The function of the ServiceGrounding is to provide sufficient information to enable some client to physically connect to, and invoke, the service: it is a mapping from an abstract to a concrete specification of the service. Currently, an OWL-S

ServiceGrounding provides a mapping to a WSDL service description (which itself uses SOAP during communication) to effect this function.

There are various initiatives under way by the OWL-S coalition to extend the OWL-S framework beyond the basic ontologies and use cases. For example, [McIlraith & Son, 2002] is an attempt to address the problem of automated Web service composition using the logic programming language Golog. High level generic procedures, for example travel planning procedures, are combined with user preferences to invoke a complex sequence of semantically marked up Web services. Golog programs and Web services communicate via the Open Agent Architecture (OAA) [Martin et al., 1999]. [Mandell & McIlraith, 2003] is an effort to demonstrate the advantages of integrating Semantic Web service technology into an industry standard effort, BPEL4WS, to enable automated service discovery, customization and semantic translation. A set of DAML-S sentences are input to the system and Web services are invoked, as required, depending on which of their DAML-S profiles are matched with the sentences.

2.5.3 The Web Service Modeling Framework (WSMF)

The Web Services Modeling Framework (WSMF) is an initiative of the SDK WSMO Group which is itself part of the SDK Cluster (www.sdk-cluster.org/). Its objective, much like that of OWL-S, is to provide the appropriate conceptual model for developing and describing Semantic Web services and their composition. Its philosophy is based on the principal of *maximal de-coupling* complemented by a *scalable mediation service* [Fensel & Bussler, 2002, Lara et al., 2003]. WSMF consists of four main elements: *ontologies* that provide the terminology used by other elements, *goal repositories* that define the problems that should be defined by Web services, *Web service descriptions* that define various aspects of a web service and *mediators* which bypass interoperability problems. Orthogonal to these four elements, WSMF has created three initiatives to develop various aspects of the framework:

- Web Service Modeling Ontology (WSMO) [WSMO, 2004] defines the ontologies for the four elements defined above. Currently, the WSMO standard presents its ontologies at a high level of abstraction, undefined for an actual implementation.

- Web Service Modeling Language (WSML) has as its objective the development of a proper formalization language for the Semantic Web services and providing a rule-based language for the Semantic Web.
- Web Services Execution Environment (WSMX) [WSMX, 2004a, WSMX, 2004b] has as its objective the creation of a reference implementation of an execution environment for WSMO. The goal is to provide both a test bed for WSMO and to demonstrate the viability of using WSMO as a means to achieve dynamic interoperability of Web services.

Although WSMF has yet to provide a practical implementation - for example it has yet to define any use cases - it remains a very promising prospect as a future fully-defined framework for the development of Web services. WSMX is a framework which OWL-S does not attempt to provide (WSMO may be compared with OWL-S) whereas, at a finer level of detail, it arguably attempts to solve certain problems OWL-S does not address. For example, effecting communication with its mediators between different ontologies and, also, WSMF is careful to point out the difference between orchestration and choreography in its process model.

2.5.4 IRS-II

IRS-II [Crubézy et al., 2001, Motta et al., 2003] originated from within the context of the IBROW project, a European Commission sponsored effort comprising a number of European partners including The Open University¹. IRS-II in particular is an effort to enable automatic configuration of reusable knowledge components, available from distributed libraries on the Internet, these libraries being structured according to the *Unified Problem-Solving Method Development Language* (UPML) framework which is illustrated in Figure 2.8.

¹ See hcs.science.uva.nl/projects/ibrow/home.html

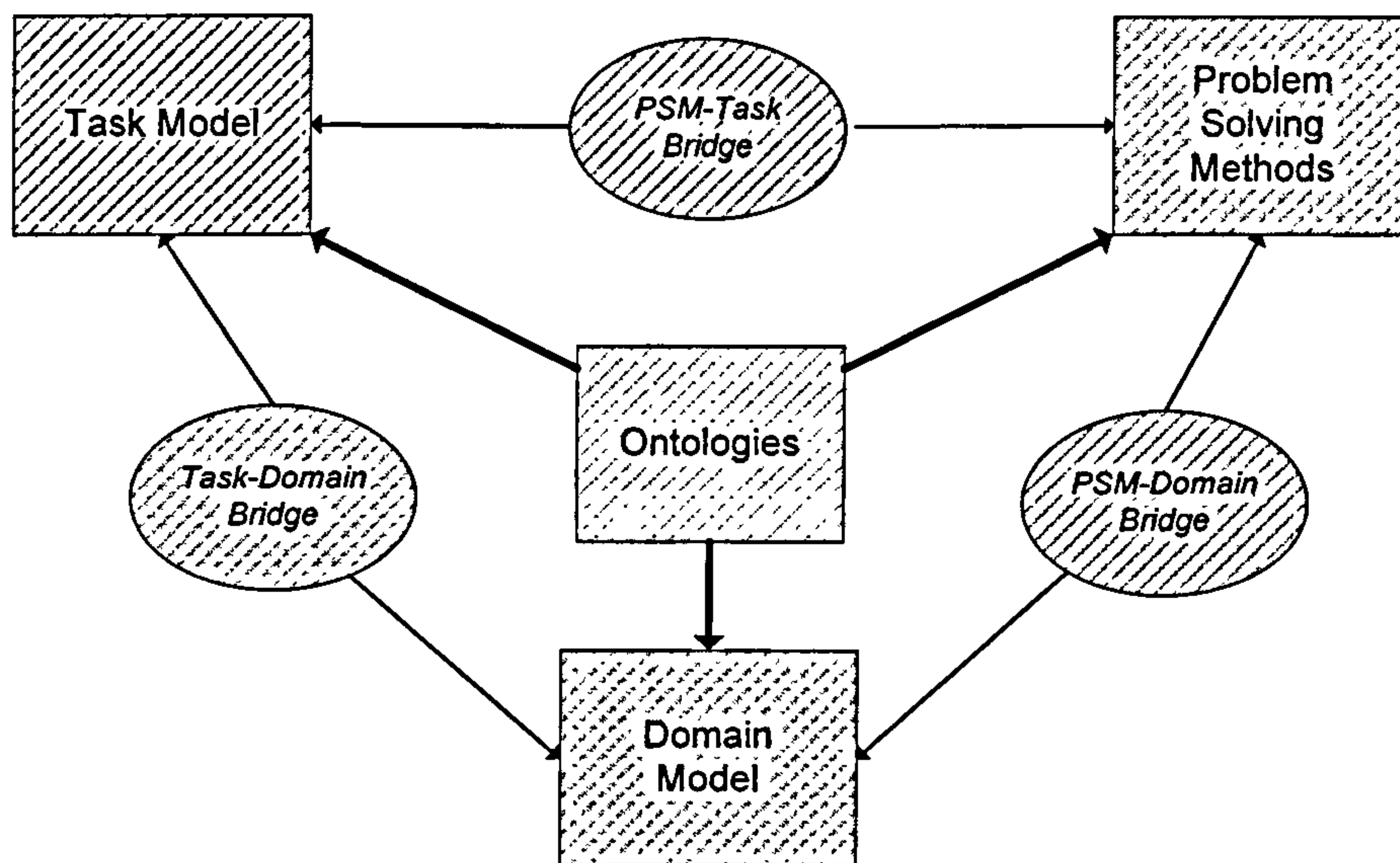


Figure 2.8: UPML Framework

The four key components in this architecture are:

- *Domain Models* which are an ontological description of the problem domain (e.g. vehicles, a medical disease).
- *Problem Solving Methods (PSMs)* [Fensel & Motta, 1998]. A semantic description in the language OCML of what some service does. The PSM description and physical service are linked via an application, the *IRS Publisher* which forwards the information to an *IRS Server* where the service is published.
- *Task Models* are what a user of the system wishes to achieve. They are a generic description of the task to be solved and may be 'high-level generic descriptions of complex classes of applications, such as classification or scheduling as well as more mundane problem specifications, such as exchange rate conversion'.
- *Bridges* map between the model components much like mediators in WSMF.

A key objective of IRS-II has been ease of use, making it relatively easy to publish and invoke Semantic Web services. Indeed, IRS-II is further down the road than OWL-S or WSMF in actually having an implemented system which users may experiment with. However, in IRS-II, services are considered atomic and there is poor functionality concerning a process model and complex choreography.

2.6 Summary

As this chapter has shown, many initiatives towards the Semantic Web are under way and it is certainly now becoming possible to develop useful frameworks and applications on top of the current technology. Indeed, the remainder of this work will demonstrate the usability and shortcomings of the OWL initiative, and associated technologies, in conjunction with the framework introduced here to make text classifier functionality useful on the Semantic Web.

Following the above discussion it follows that there may be considered to be two key efforts to implement a Semantic Web: one which develops a framework on top of OWL and the WSMO initiative. Of these, OWL and efforts surrounding it have achieved a higher level of maturity as a set of technologies amenable to implementation. OWL itself is a W3C recommendation and OWL-S, as a services framework, is reaching a stage where services may be invoked from their OWL-S descriptions (consider in particular OWL-S API from the University of Maryland¹). WSMO, on the other hand, is a top down design. It is, first and foremost, an effort towards Semantic Web services and the ontology begins with the design of the high level system components (ontologies, services, mediators and goals) and is working towards, but has not yet achieved, an implementation. It is for reasons of ease of implementation, therefore, that OWL is adopted for practical use in this thesis.

¹ See www.mindswap.org/2004/owl-s/api/

Chapter 3 Machine Learning and Text Classification

3.1 Introduction

Machine Learning is the study of computer algorithms that improve automatically through experience. Text classification is one particular application domain of machine learning that uses some inductive algorithm to construct a classifier from a classified corpus of documents which can be used to suggest classifications for previously unseen documents. This chapter explains the key stages required to build such a classifier, in particular text preparation, feature extraction and classifier construction and it then describes in some detail two specific classifier technologies, namely Naïve Bayes, a probabilistic classifier, and Support Vector Machines (SVMs).

3.2 Text Classifiers

A key motivation for the work in this thesis has been the automatic generation of classification metadata for unseen documents. In particular, the ACM Computing Classification System was targeted as a particularly valuable scheme when considered in conjunction with the AKT project. However, this is only one of a number of possible classification schemes which may be considered and indeed on the Semantic Web we would like to be able to build classifiers for any arbitrary classification scheme. As a consequence of this, machine learning techniques are ideally suited because they do not require the involvement of domain experts to set up a system, yet they have been shown

to provide classification accuracies comparable to those of other candidate classification methods, such as creating a rule based system based on knowledge engineering techniques or straightforward manual classification.

Indeed, inductive construction of classifiers has been achieved in various ways using techniques such as probabilistic classifiers, decision trees, regression methods and neural networks. An in-depth presentation of this research area is presented in [Sebastiani, 1999]. Whichever technique is selected, the principle is the same. A corpus of documents, previously classified according to some scheme, is required to train a classifier which may then be called upon to attempt to classify a previously unseen document according to the given classification scheme.

From this discussion, a number of questions arise. Which classifier is best for which corpus? What are the properties of a corpus which make it the best choice for some specific algorithm? What are the performance constraints? Answering these questions properly may be beyond the scope of this thesis: there are simply too many parameters and algorithms. However, by considering a couple of chosen classifier technologies, a picture may begin to emerge.

The construction of an effective classifier can be considered to comprise three components, namely text preparation, feature selection and actual creation of the classifier itself. Each of these components is considered in turn below. Two approaches to classifier creation have been adopted which will be developed more fully in a software framework. These are Naïve Bayes, a probabilistic classifier which is quite simple and which is employed to provide some baseline results and Support Vector Machines, a recent development known to produce quality performance.

3.3 Text Preparation

The classification techniques described below make use of the well-known concept of considering a given document as a ‘bag of words’. That is, classification relies on the presence or absence of particular words in a document and, optionally, the frequency of occurrence of those words. No value is placed on, for example, the ordering or context of any given word.

Like many repositories of technical documents, the ACM manages the vast majority of its documents in pdf format. The first stage of text preparation is consequently to convert each of these documents to raw text. A library which was free until recently has been used to perform this process¹.

Two procedures, namely stemming and stopwords removal, are then performed on the raw text. These processes are carried out on the raw text prior to both feature selection and classifier creation and use.

Stemming is based on the principle that morphological variations of a particular word can be considered to have the same semantic interpretation and consequently may be reduced to a single form. For example, the words compute, computation and computing may be reduced to comput. The well-known Porter's stemming algorithm has been used in this work².

Stopword removal involves the removal of certain frequently occurring words (maintained in a list) from the raw text. These words can be considered to have a low discrimination value across the document corpus, because they are evenly distributed throughout the documents. Example stopwords are 'and', 'the' and 'only'.

The main objective of this pre-processing is to reduce the dictionary size during the next stages of text classification, consequently reducing memory requirements and processor time. Stopword removal is common in the field of information retrieval. Whether stemming is used or not has been shown typically to make little difference to the quality of results derived from text classification.

3.4 Feature Selection

The feature set, or vector, is a set of words carefully selected to represent words which are common across all the classes being considered for classification. The objective is that these words will enable discrimination between classes when incorporated into a classifier. They typically reflect the nature of the classes under examination so, for example, if a class is hardware, you might find words such as processor, transistor or chip represented in the feature vector. So, the feature vector is a carefully chosen subset

¹ See www.jpedal.org

² See www.tartarus.org/~martin/PorterStemmer/

of the total number of words available. There are good reasons for this *dimensionality reduction*. First, it has been observed in practice that if all the words are used, *overfitting* results. Classifiers which overfit are good at classifying documents they have already seen, but not so good when it comes to completely new documents. Second, when raw text is extracted from a format such as PDF, the process often produces a lot of garbage e.g. word strings made up of corrupted characters or simply concatenated strings. Filtering is preferred to remove these cases.

Many techniques are available for feature selection. Again, [Sebastiani, 1999] provides a good introduction to the subject and [Rogati & Yang, 2002] give an illustration of how well various methods perform. In particular, *information gain* is a popular and effective method and is the one selected for use here. Equation (1) shows the function for evaluating information gain:

$$IG(x_k, c_i) = P(x_k, c_i) \log \frac{P(x_k, c_i)}{P(c_i).P(x_k)} + P(\bar{x}_k, c_i) \log \frac{P(\bar{x}_k, c_i)}{P(c_i).P(\bar{x}_k)} \quad (1)$$

In practice, a feature set is assembled as follows. Counts are made of each feature x_k using raw text from ACM originated documents for which a class c_i already exists. So, for example, $P(x_k, c_i)$ is the probability that word x_k is randomly selected from all the words in class c_i and $P(\bar{x}_k, c_i)$ is the probability that any word other than x_k is randomly selected from all the words in class c_i . From these counts, the probabilities in (1) may be estimated. Information gain is calculated for each word belonging to each class and then for each word, the highest of its values of information gain is taken. Words are sorted by their (now single highest) gain value and from this list the feature set is extracted.

3.5 Naïve Bayes

Bayes' rule as applied to text classification [Lewis, 1998] is:

$$P(c_k | \mathbf{x}) = P(c_k) \times \frac{P(\mathbf{x} | c_k)}{P(\mathbf{x})} \quad (2)$$

where c_k is the k th class from a set of classes to be considered and $\mathbf{x} = (x_1, \dots, x_j, \dots, x_d)$ is a feature vector created from a text file. For a new document to be classified, Naïve Bayes works by evaluating the probability of each possible class given the observed feature vector, that is $P(c_k | \mathbf{x})$, and classifying the document as belonging to the class with the highest evaluated probability.

To build and evaluate a classifier, a corpus of manually classified documents is required which has representatives of each of the classes that the classifier is to consider. This corpus is then divided into a *training set*, with which the classifier is built, and a *test set* to evaluate its effectiveness. The training set is required in order to evaluate $P(c_k)$ and $P(\mathbf{x} | c_k)$. $P(\mathbf{x})$ is a constant for a given file during this process and may be ignored. $P(c_k)$ is a straightforward fraction of the number of document occurrences for each class divided by the total number of documents. On the other hand, many techniques are available for evaluating $P(\mathbf{x} | c_k)$ [McCallum & Nigam, 1998, Duda & Hart, 1973, Eyheramendy et al., 2003, John & Langley, 1995]. The work carried out here considers two of these techniques, the *multinomial* and *multivariate* event models which are considered below.

In general, because it is assumed that the occurrence of a particular value of x_j is statistically independent of the occurrence of any other $x_{j'}$ (what is known as the Bayes' independence assumption), $P(\mathbf{x} | c_k)$ in (2) can be written as:

$$P(\mathbf{x} | c_k) = \prod_{j=1}^n P(x_j | c_k) \quad (3)$$

This result is used in what follows.

3.5.1 Multinomial Naïve Bayes

Multinomial Naïve Bayes makes use of the frequency of words in documents to produce its results. This may be seen by considering the multinomial distribution in (4).

$$P(n_1, \dots, n_J) = \binom{N}{n} \prod_{j=1}^J p_j^{n_j} \quad (4)$$

The multinomial distribution may be considered an extension to the binomial distribution where there are more than 2 independent random variables ($J > 2$), for example throwing a die. Instead of the faces of a die, the multinomial distribution considers that each random variable is one of the J words which appear in the feature set. Indeed, think of the words being in a bag and the probability p_j of picking out a specific word is then the same as the probability of randomly coming across that word in a document of that class. Assume that, for a given document, you find n_j (from (4)) occurrences of the j th feature in the document which is being classified. p_j is therefore the same as $P(x_j | c_k)$ and evaluating the multinomial distribution is one particular way of solving (3). The multinomial coefficient $\binom{N}{n}$ which is a shorthand for $N! / \prod_{j=1}^J n_j!$ (where N is the length of the document, i.e. the number of features it contains) is a constant for a given document across all classes and may be ignored. p_j for each word within each class is calculated from the training set according to (5). Evaluating (4) as if it belonged to each class in turn results in a figure which may be ranked to indicate the likelihood of the document belonging to a specific class.

$$p_j = P(x_j | c_k) = \frac{1 + \sum_{i=1}^{|D|} N_{ji} P(c_k | \mathbf{x}_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{is} P(c_k | \mathbf{x}_i)} \quad (5)$$

In (5), $|V|$ is the number of features, $|D|$ is the number of documents in the training corpus, N_{ji} is the count of the number of times word w_j occurs in document d_i and $P(c_k | \mathbf{x}_i) \in \{0,1\}$. If there are no word occurrences for a given feature in a class, (5) defaults to $1/|V|$. This *Laplace smoothing* ensures all features are considered in classifier

construction and if a feature has a frequency of zero for a given class, its contribution is a very small value for $P(x_j | c_k)$.

3.5.2 Multivariate Naïve Bayes

Multivariate Naïve Bayes does not make use of the frequency of features in either the training or test set of documents to build and use its classifiers. For a feature vector $\mathbf{x}$, $x_j = \{0,1\}$ is dependent simply on whether or not a feature is present in a document.

To classify a document, (6) is evaluated for each possible class.

$$P(\mathbf{x}_i | c_k) = \prod_{j=1}^{|V|} (B_{ij} P(x_j | c_k) + (1 - B_{ij})(1 - P(x_j | c_k))) \quad (6)$$

$B_{ij} = \{0,1\}$ and indicates whether or not a particular feature appears in the document being classified. The training set is required to assemble values for $P(x_j | c_k)$ and is achieved by making use of (7).

$$P(x_j | c_k) = \frac{1 + \sum_{i=1}^{|D|} B_{ij} P(c_k | \mathbf{x}_i)}{2 + \sum_{i=1}^{|D|} P(c_k | \mathbf{x}_i)} \quad (7)$$

$P(c_k | \mathbf{x}_i) = \{0,1\}$ and Laplace smoothing is again used, this time to avoid probabilities of 0 or 1.

3.6 Support Vector Machines

3.6.1 Introduction

Recall that a document may be represented by its feature vector $\mathbf{x} = (x_1, \dots, x_j, \dots, x_d)$ where x_j is the count of the j th feature in the document. Support Vector machines (SVMs) consider such a document to be represented in geometrical terms as a point in d dimensional space and, for a set of such points belonging to each of 2 classes, SVMs use techniques to find a $d - 1$ dimensional hyperplane which separates the two sets with

the largest possible margin. In recent work [Sebastiani, 1999, Dumais et al., 1998], Support Vector Machines have been shown to be a highly effective technique for text classification, outperforming other methods such as Naïve Bayes.

Using Support Vector Machines to perform text classification is a fairly recent innovation introduced by Joachims [Joachims, 1998]. What follows is a gentle introduction describing their operation. For further information, the reader is referred to [Cristianini & Shawe-Taylor, 2000, Schölkopf, 2000, Chen et al., 2003, Cortes & Vapnik, 1995].

3.6.2 The Simplest Support Vector Machine

Consider the classification of two sets of points which are linearly separable in the feature space. Figure 3.1 illustrates this situation with the two sets represented by circles and squares. Next consider the problem of finding the plane which separates the two points with the maximum margin of separation. The *maximal margin classifier*, which is now described, is the simplest formulation of a Support Vector Machine that can find a solution in this situation.

It can be seen in Figure 3.1 that the plane separating the two sets of points is bounded by a margin on each side. In this example, the margin may be seen to be bounded by three points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$. These points are the *support vectors* and in fact they are the only points required in order to define the separating hyperplane. $\mathbf{w}$ is a vector which is orthogonal to the hyperplane. The margin may be calculated by projecting the vector connecting two support vectors, here $\mathbf{x}_1$ and $\mathbf{x}_2$, onto $\mathbf{w}$ following (8).

$$margin = \frac{1}{2} \frac{1}{\|\mathbf{w}\|} \mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) \tag{8}$$

Now consider the two vectors which define the margins:

$$\begin{aligned} (\mathbf{w} \cdot \mathbf{x}_1) + b &= +1 \\ (\mathbf{w} \cdot \mathbf{x}_2) + b &= -1 \end{aligned}$$

so that

$$(\mathbf{w} \cdot \mathbf{x}_1 + b) - (\mathbf{w} \cdot \mathbf{x}_2 + b) = 2$$

$$\Rightarrow \mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2$$

and substituting this result back into (8) yields:

$$\text{margin} = \frac{1}{\|\mathbf{w}\|} \quad (9)$$

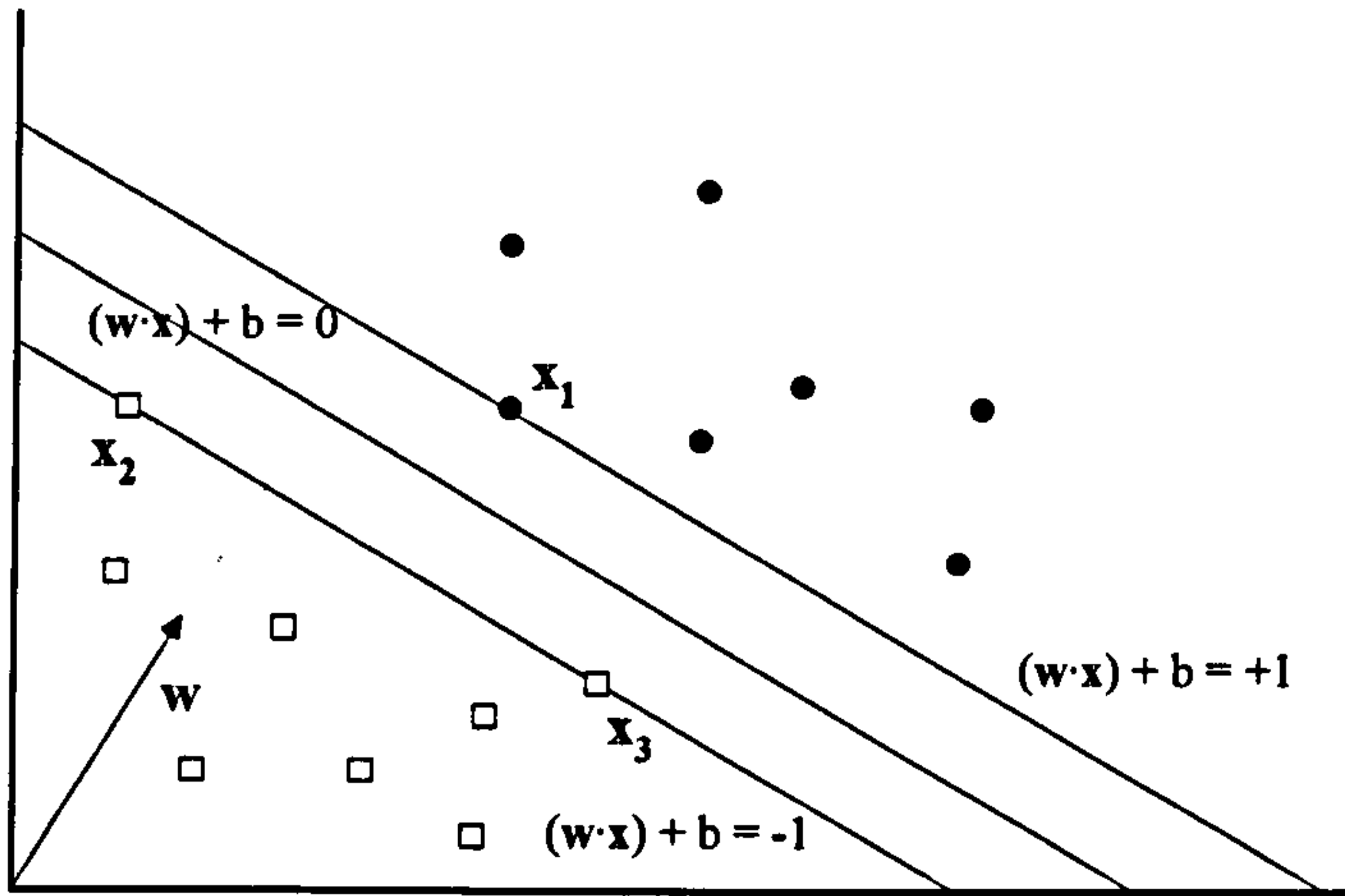


Figure 3.1: Two Classes Separated by a Plane with Maximal Margin

The result for the margin of the maximal margin classifier given by (9) leads directly to the result demonstrated by Proposition 3.1.

Given a linearly separable training sample

$$S = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l))$$

the hyperplane $(\mathbf{w}, b)$ that solves the optimisation problem

$$\begin{aligned} &\text{minimise} && (\mathbf{w} \cdot \mathbf{w}) \\ &\text{subject to} && y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \\ &&& i = 1, \dots, l \end{aligned}$$

realises the maximal margin hyperplane with geometric margin $1/\|\mathbf{w}\|$

Proposition 3.1: Optimisation Problem for the Maximal Margin Classifier

A solution to this problem may be found by recourse to *Lagrangian optimisation theory* in which the above problem may be expressed as

$$\max_{\alpha \geq 0} (\min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)) \quad (10)$$

where

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) - \sum_{i=1}^l \alpha_i (y_i ((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1) \quad (11)$$

$$\alpha_i \geq 0$$

The Lagrangian L has to be minimised with respect to the *primal variables* $\mathbf{w}$ and b and maximised with respect to the *dual variables* α_i . Using this result, stationary points may be found:

$$\begin{aligned} \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^l y_i \alpha_i \mathbf{x}_i = 0 \\ \Rightarrow \mathbf{w} &= \sum_{i=1}^l y_i \alpha_i \mathbf{x}_i \end{aligned} \quad (12)$$

$$\begin{aligned} \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} &= \sum_{i=1}^l y_i \alpha_i = 0 \\ \Rightarrow 0 &= \sum_{i=1}^l y_i \alpha_i \end{aligned} \quad (13)$$

The first of these relations shows that the solution vector $\mathbf{w}$ can be expressed as a linear combination of the training points. Substituting these results into the primal problem of (11) yields the following result:

$$\text{maximise:} \quad L(\mathbf{w}, b, \alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad (14)$$

$$\begin{aligned} \text{subject to:} \quad & \sum_{i=1}^l y_i \alpha_i = 0 \\ & \alpha_i \geq 0 \\ & i = 1, \dots, l \end{aligned}$$

Note that this dual problem requires the solution to a number of variables (α_i) equal to the size of the number of examples in the training set of documents. This is, therefore, a simpler problem to solve than the primal one where $\mathbf{w}$ and b must additionally be found.

3.6.3 Allowing for Imperfect Data

The maximal margin classifier assumes that the training data is linearly separable. Unfortunately this is rarely possible in practice and certainly not the case in text classification where manually classifying documents is not a precise science. Manually classifying documents is a question of judgement. Offer the same document to two different experts and it is quite possible they will decide on different classifications. In the terminology of Figure 3.1, boxes may be found with circles and vice versa.

In order to find a solution to this issue, the notion of *slack variables* (ξ) has been introduced which allow the margin constraints of Proposition 3.1 to be violated [Cortes & Vapnik, 1995]. The optimisation problem consequently becomes:

$$\begin{aligned} \text{minimise} \quad & \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + C \sum_{i=1}^l \xi_i \\ \text{subject to} \quad & y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \\ & i = 1, \dots, l \end{aligned} \tag{15}$$

This has a primal Lagrangian of the form:

$$L(\mathbf{w}, b, \xi, \alpha, \mathbf{r}) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + C \sum_{i=1}^l \xi_i - \sum_{i=1}^l \alpha_i (y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1 + \xi_i) - \sum_{i=1}^l r_i \xi_i \tag{16}$$

which leads to the following results:

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \mathbf{r})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^l y_i \alpha_i \mathbf{x}_i = 0 \tag{17}$$

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \mathbf{r})}{\partial \xi_i} = C - \alpha_i - r_i = 0 \tag{18}$$

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \mathbf{r})}{\partial b} = \sum_{i=1}^l y_i \alpha_i = 0 \tag{19}$$

and substituting these results back into the primal yields the following result:

$$\begin{aligned}
&\text{maximise:} && W(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) && (20) \\
&\text{subject to:} && \sum_{i=1}^l y_i \alpha_i = 0 \\
&\text{where:} && C \geq \alpha_i \geq 0 \\
&&& i = 1, \dots, l
\end{aligned}$$

It may be observed that this is almost the same as the dual problem in (14) for the maximal margin classifier with the exception of the bound on C . In practice, a good approximation to an optimal value of C is found by solving the dual problem through a range of C and taking the best result.

3.6.4 Kernels

In the description of Support Vector Machines outlined above, the relatedness of two vectors in feature space i.e. the likelihood that they belong to the same class, is calculated by a simple dot (or scalar) product of the two feature vectors, that is $(\mathbf{x}_1 \cdot \mathbf{x}_2)$. This assumes that the feature vectors are related by a simple linear combination of their component features. In practice, a more complicated relationship may exist and there is a benefit to mapping the feature space into a higher dimensional, possibly infinite dimensional, space:

$$\mathbf{x} = (x_1, \dots, x_n) \mapsto \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_N(\mathbf{x})) \quad (21)$$

where N may be very much larger than n . The difficulty with this is that mapping into such a high dimensional space makes the computational task of solving the dual problem prohibitively expensive. Fortunately kernel functions exist which achieve the same result by effectively using some function to transform the result of the dot product mapping in the original feature space.

As an example, consider the following mapping from a 2-dimensional to a 3-dimensional space:

$$\phi(\mathbf{x}) = (x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2) \quad (22)$$

Now let us consider using this mapping as a kernel function:

$$\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= \phi(\mathbf{x}) \cdot \phi(\mathbf{x}') \\
&= x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_2 x_1' x_2' \\
&= (\mathbf{x} \cdot \mathbf{x}')^2
\end{aligned} \tag{23}$$

In this way mapping into the higher 3-dimensional space can be avoided simply by taking the square of the dot product of the original feature vectors.

In fact, many less trivial kernel functions exist. Four of the most common are listed below:

- linear: $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}') \tag{24}$

- polynomial: $k(\mathbf{x}, \mathbf{x}') = (\gamma \mathbf{x} \cdot \mathbf{x}' + r)^d, \gamma > 0 \tag{25}$

- radial basis function (RBF): $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2), \gamma > 0 \tag{26}$

- sigmoid: $k(\mathbf{x}, \mathbf{x}') = \tanh(\gamma \mathbf{x} \cdot \mathbf{x}' + r) \tag{27}$

Indeed (23) may be considered as one particular instance of (25). Also, (24) is the default case on which the description of Support Vector Machines up to this section have been based.

3.6.5 Solving the Dual Problem

Finding a solution to a classification problem with Support Vector Machines involves solving (20) for all α_i and substituting these values into (17) for $\mathbf{w}$ and the primal constraints (Proposition 2.1) for b . Unfortunately, although a number of quadratic optimisation techniques are available which can solve this problem, they typically require a matrix to be held in memory (of $y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$) whose size increases as the square of the number of training examples. The size of this matrix crucially limits the size of the optimization problem which may be solved, due to the limitations imposed by the amount of available fast memory.

To overcome memory constraints, a number of algorithms have been introduced which attempt to use a smaller working set of the training data. *Chunking* solves the optimisation problem for a subset of the training data and iteratively replaces those examples found not to be support vectors with samples from the to-date unseen training

corpus. *Decomposition* guarantees a fixed memory requirement by limiting the working set to a fixed size. Training examples are swapped in and out of the set as required.

Sequential Minimum Optimization (SMO) is arguably the most widely used technique at present and it can be considered as an example of the decomposition algorithm in which the working set is maintained at the minimum number of two training examples. SMO was introduced by Platt and is explained in detail in [Platt, 1998]. At each iteration, two training samples are selected according to some heuristic and optimised analytically while maintaining the condition (19). The result of each iteration is an increase in the value of the dual objective function and the algorithm continues until some stopping criterion is met (for example, stop after the increase in the dual objective is consistently less than some limit).

3.6.6 Multi-Class SVMs

Clearly, a limit to the use of support vector machines as outlined above is simply the observation that they only classify two classes. To extend their use to more than two classes, a number of techniques have been put forward. These techniques may be divided into two groups:

1. Techniques which consider solving an optimisation problem in one step by generalising (15) to handle all the classes of interest [Weston & Watkins, 1999].
2. Techniques which combine a number of binary classifiers in some way and typically make use of a voting strategy to select the result of a classification.

Three such methods are commonly used: *one-against-all*, *one-against-one* and *DAGSVM*.

Suppose that there are n classes in the classification problem. One-against-all builds n classifiers each of which is a binary classifier consisting of one of the classes and a representation of training examples from all the other classes. One-against-one builds $n(n-1)/2$ binary classifiers where each and every class is classified against all the other classes. DAGSVM [Platt et al., 2000] builds a directed acyclic graph of $n(n-1)/2$ classifiers which reduces the number of binary classifiers over one-against-one required to classify a document.

[Hsu & Lin, 2002] is an effort which has empirically compared these various methods. It concludes that one-against-one and DAGSVM are generally better suited for large classification problems. In addition, classification of documents according to the ACM hierarchy has a requirement that multiple candidate classes may be attributed to a given document. For example, a document may be ascribed to both the ‘D. Data’ and ‘E. Software’ categories. Since DAGSVM does not enable this kind of multiple classification, the one-against-one approach, in which a threshold of votes is used as the decision criterion, was in practice implemented in the work undertaken here.

3.7 Summary

This chapter has provided some theory behind text classification in general and two specific classifier algorithms in particular. Naïve Bayes is a well known algorithm which is simple to implement, computationally inexpensive to run and should provide a baseline idea of performance. Support Vector Machines are the opposite: algorithmically complex but, according to the literature, providing top-notch performance. Although many different technologies may be considered, this chapter has focused on those which are actually implemented in this work. The next chapter deals with this practical implementation. It will introduce an extensible software framework which incorporates these classifier algorithms and continues with an outline of their performance using two corpora: the ACM corpus of Computer Science related documents and the readily available Reuters-21578 corpus of newswire articles.

Chapter 4 Classifier Performance

4.1 Introduction

The previous chapter introduced some theory behind machine learning applied to text classification in general and for two specific approaches in particular, namely Naïve Bayes and Support Vector Machines (SVMs). One motivation for the work in this thesis is to establish some idea of the effectiveness of text classifiers in the real world: how well do they perform and to what extent may they be relied upon. This chapter introduces a flexible software framework which allows various classifier algorithms to be tested and discusses performance for the two classifier technologies selected in relation to two tagged corpora of documents, namely the Reuters-21578 corpus and the ACM corpus.

4.2 A Software Framework

4.2.1 Background

Ultimately, the objective of this work is to enable the automatic generation of classifiers via a set Web services enabled for the Semantic Web. As a stepping stone to that point, a flexible software framework is required that facilitates the incorporation of various classifier algorithms. This framework can then itself be tested and developed before further transformation to Web services. Additionally, in a standalone application, exhaustive performance tests may be made of the classifier algorithms. Classifiers have processor-intensive requirements and we wish to avoid the inevitable effect on program

execution time that inevitably results from transmitting all communication over a network.

With this in mind, two stand alone applications were developed with Microsoft Visual Studio .NET on a Windows XP platform. Firstly, *Features* is an application which is written in C# and performs feature extraction given a tagged corpus. This employs an algorithm based on information gain which was previously discussed in section 3.3. Secondly, *Classifiers* is an application written in C++ which performs classifier creation and testing given a feature set and a tagged corpus.

Two corpora have been employed for testing purposes. Reuters-21578 has been widely used in Classifier research and is readily available for Internet download¹. It was compiled by David Lewis and consists of 21,578 Reuters newswire articles from 1987. The ACM corpus was assembled for use in this work using scripts written in the Perl programming language which were employed to ‘screen scrape’ the ACM document portal^{2,3}. This work therefore introduces the ACM corpus as an empirical test bed for text classifier research. It consists of over 119,000 PDF documents relating to Computer Science research.

It may be noted that these two corpora are quite different in their textual properties. Reuters-21578 documents are raw text and have a maximum document length of about 60 tokens when considering 10,000 features. On the other hand, ACM documents have raw text extracted from PDF documents and have an average document length of about 1,300 tokens when considering 14,000 features. (These token numbers refer to the number of stopped and stemmed words which match features when a large feature set is considered). In addition, Reuters-21578 is known to have a close correspondence between discriminating words and their associated classes. For example, appearance of the word ‘wheat’ is a good predictor that the article is associated with the class ‘wheat’. This is not the case with the ACM corpus where classes tend to cover a much larger range of topic areas and words which match features may be considered to be much more thinly spread. For example, one top level ACM class is ‘Hardware’. It may

¹ See, for example, <http://www.daviddlewis.com/resources/testcollections/reuters21578/>

² See portal.acm.org

³ See Chapter 5. ClassAKT was in fact developed before the software framework discussed here and the ACM corpus was assembled in conjunction with that.

be considered that pertinent features might include words such as ‘transistor’ or ‘chip’ but it is still however possible that some documents associated with this class contain neither of these words.

4.2.2 A Software Framework for Classifiers

The development of ClassAKT described in Chapter 5 actually predates the software architecture introduced here. It uses the Weka classifier library¹ for its implementation of the Naïve Bayes algorithm. It was then decided that for future work, a new software framework, simply called *Classifiers*, written in C++, would be created and this is discussed here. This was because arguably the most popular Support Vector Machine implementations (SVM-Light by Joachims and LIBSVM from the National Taiwan University) were available in C++. Also, a number of performance and reliability issues were observed in the Java environment in which Weka was employed on the very large ACM dataset. From personal experience of the author in developing processor intensive applications, it was felt that using the C++ language was likely to result in a more robust framework.

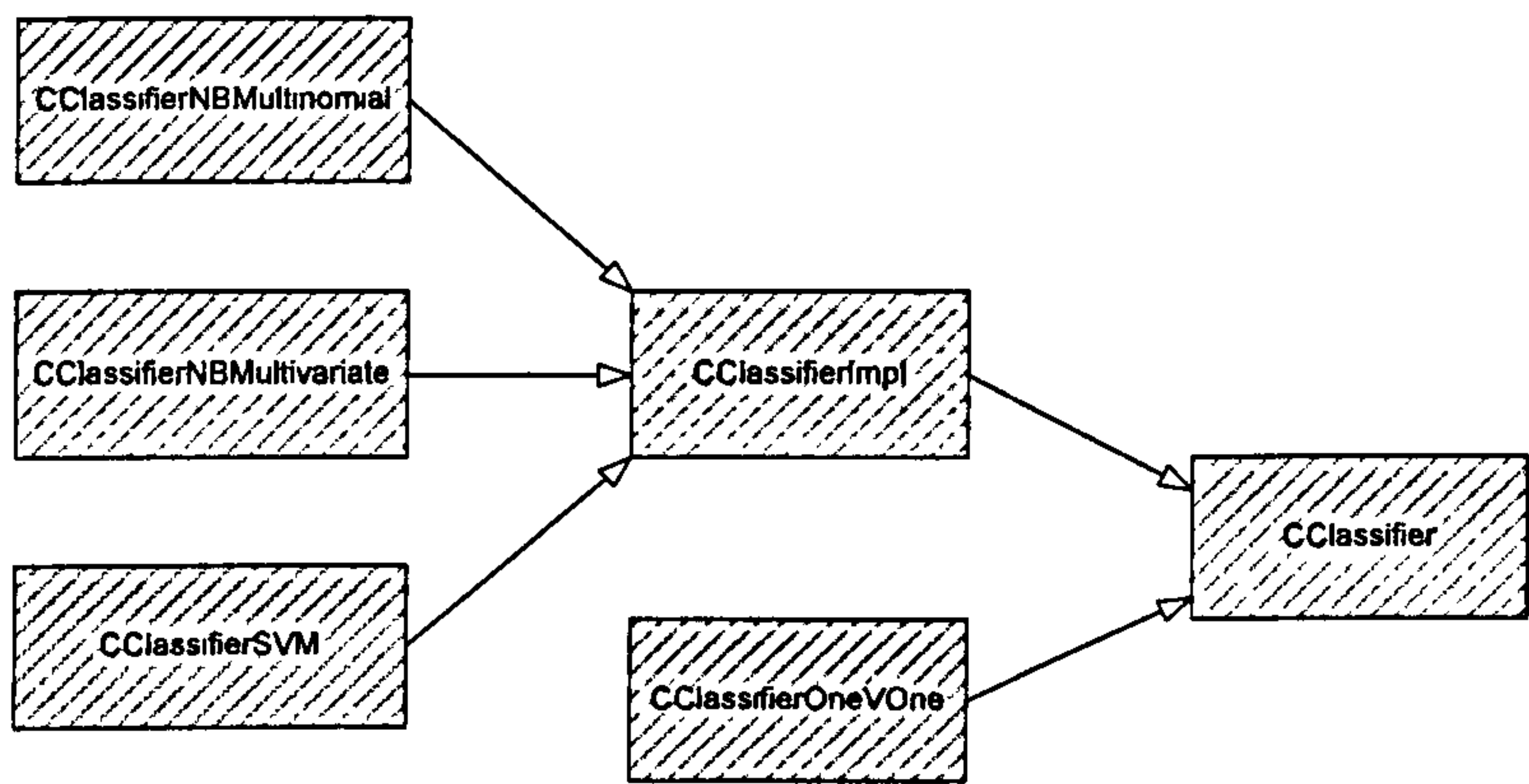


Figure 4.1 Class Hierarchy for Classifiers Architecture

Figure 4.1 shows the class hierarchy for the *Classifiers* architecture. *CClassifierNBMultinomial* and *CClassifierNBMultivariate* implement multinomial and

¹ See <http://www.cs.waikato.ac.nz/ml/weka/>

multivariate Naïve Bayes respectively. Their implementations are derived directly from the algorithm descriptions in Chapter 3. *CClassifierSVM* is a SVM implementation, or more correctly, a wrapper around the LIBSVM SVM implementation. LIBSVM [Chang & Lin, 2004, Hsu et al., 2003] is a library produced by a team at the University of Taiwan and is freely available for download¹.

Of course, a SVM is a binary classifier, capable of distinguishing between just two classes at a time. *CClassifierOneVOne* is a ‘manager’ class which manages instances of binary classifiers to assemble a classifier which can classify two or more classes using the one-against-one described in section 3.6.6. Although it can handle binary classifiers of any type (for example it could assemble a committee of binary Naïve Bayes classifiers), its primary purpose is to coordinate a number of binary SVM classifiers so that SVMs may be employed directly to classify the multiple classes of most real situations such as are found in ACM or Reuters-21578.

To paint a fuller picture of the architecture, it is perhaps useful to briefly describe the principal configuration files, all structured in XML, which exist in the system:

- *Info files* describe either a part corpus or a full corpus. They essentially map the classes of documents to the documents themselves, maintained in a sub-directory as raw text files.
- *Features files* enumerate a set of features and associate with them the classes for which they were evaluated.
- *Classifier files* associate a classifier specification (for example SVM utilising a one-v-one committee style) with a set of feature files which are required to establish the classifier. So, for example, a typical Naïve Bayes classifier file will contain a single features file name which was established for all the classes under consideration whereas the SVM committee approach, where there are n classes, will contain a list of $n(n-1)/2$ features file names.
- *Batch files* enable the testing of classifiers. They enumerate a list of specifications for testing. Each specification contains a classifier file name, the number of files to be used to build the classifier, the number of features to be used to build the classifier, the proportion of files to be used for testing and the number of times to

¹ See www.csie.ntu.edu.tw/~cjlin/libsvm/

run the test. So, for instance, a Naïve Bayes classifier may be build and tested 5 times each time employing 3000 features and requiring 5000 files per build and these 5000 files may comprise, say, 85% of the total files used, the remaining 15% being employed to create a set of test results.

The results that follow in this chapter were assembled from running various batch files in turn keeping a fixed number of features and varying the number of files and then vice versa, varying the number of files for a fixed number of features. The fixed number of files or features were numbers which intuitively were felt would give a sensible, average classifier performance. The *Classifiers* program is able to produce an extensive log file when configured to run a batch file. This log file includes a line for each test file indicating its manual result together with the result predicted by the classifier. *Classifiers* also evaluates and outputs averages for precision and recall taken across all of the runs for each configuration of a classifier build. Measures of performance, in particular precision and recall, are described in the next section before some actual performance results are introduced.

4.3 Performance Measures

In common with the majority of information retrieval systems, measures of *precision* and *recall* are taken as the basis of measuring the effectiveness of a particular classifier [Sebastiani, 1999, Lewis, 1991]. In addition, measures of *accuracy* are made. With reference to Table 4.1, values of TP (*true positives*), TN (*true negatives*), FP (*false positives*) and FN (*false negatives*) are calculated as sums of classification results observed over test sets of documents, either for a particular class or globally over a set of classes. From this follows the definitions of precision, recall and accuracy.

		Classifier Decision	
		yes	no
Manual Decision	yes	TP	FN
	no	FP	TN

Table 4.1: Contingency Table

$$precision = \frac{TP}{(TP + FP)} \quad (1)$$

$$recall = \frac{TP}{(TP + FN)} \quad (2)$$

$$accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (3)$$

In general, the better the classifier, the higher the values of these measures. If precision goes up recall would be expected to go down. A low recall risks the possibility of losing correct classifications whereas if precision is low, although the correct class is more likely to be picked out, so are a number of extra incorrect results. A trade off in values of precision and recall is therefore desired and in particular, a break-even point is sought for precision and recall where both values are high and very close to each other.

Counts of TN are usually much larger than those of TP, FN and FP. For example, consider a classification over a range of 10 classes. If a single result from a classifier correctly matches the manually evaluated class then TP will be 1 and TN will be 9. Consequently, TN tends to swamp estimates of accuracy, making variations of this estimate small and any measures of accuracy less valuable.

4.4 Reuters-21578

As mentioned previously, the Reuters-21578 corpus has been widely used in classifier research. Sebastiani [Sebastiani, 1999] in particular has made a comparative study of results reported by various researchers. The Reuters test set is itself formatted into a number of splits of training/testing documents so that results of new research may be directly compared against previously published data. In particular, the ModApté split divides the corpus into 9,603 training and 3,299 testing documents spread across the 90 classes. A subset of this split is often employed in which only the 10 most commonly occurring classes are considered for classification purposes. This is the split chosen for the results prepared in this work. It is also the split selected by Joachims [Joachims, 1998] in his seminal work on SVMs in text classification so this choice permits a degree of comparison with results reported there. It should be noted that the *Classifiers* program

selects a training/test set split entirely randomly and does not work to the configuration specified in ModApté.

Figures 4.2 and 4.3 show some results employing multinomial Naïve Bayes. Multinomial Bayes may be a relatively simple algorithm but the graphs indicate that it shows quite good performance as a ‘baseline’ classifier. In Figure 4.2, the number of features is kept constant (at 5000) and a varying number of files is employed in each classifier build. In Figure 4.3, on the other hand, the number of files is kept constant (at 4000) and the number of features is varied. The results imply that the multivariate Bayes classifier is relatively insensitive to the number of features used, the graph being fairly flat, whereas a threshold of (perhaps 2500) files leads to fairly even performance. In summary, good performance by this classifier with a sensible configuration resulted in a precision and recall of around 0.9 and 0.84 respectively.

Figures 4.4 and 4.5 are similar graphs for the multivariate variation of the Bayes implementation. As can be seen, performance was not as good as for its multinomial counterpart. In this case, the classifier build was much more sensitive to the number of files employed - the larger number the better - whereas multivariate was again relatively insensitive to the number of features used.

In both versions of the Bayes classifier, the value of recall is always observed to be less than the value of precision at the same point in the graph, and indeed an actual breakeven point was difficult to establish. The Reuters corpus is, like the ACM corpus, a multi class problem inasmuch as a given document may be awarded two or more classes. The classifiers, on the other hand, were found invariably to select only a single class even when an expert had categorised a document as belonging to two. Consequently, a second class was less likely to be spotted by the classifier with the effect that recall suffers. (Intuitively, at the core of the Bayes algorithm fractions are multiplied together for each observed token and a fractional product is calculated for each class. It is felt that this multiplication process is too discriminatory and one product tends to end up being much larger than the others).

Figures 4.7 and 4.8 show some results observed with SVM classifiers (using a linear kernel). As previously described, a SVM classifier is a committee of binary classifiers. There are 10 classes utilised from Reuters. Consequently, an SVM classifier is

made up of 45 binary classifiers and in effect each class can be awarded up to 9 ‘votes’ because it is involved in 9 binary classifiers. So, if a document does belong to one class, and if the classifier works well, the correct class is awarded 9 votes and one would hope the other 36 votes are fairly evenly distributed among the other classes, thus providing good discrimination. However, in practice, there is a much flatter normalised distribution of weights between classes, a distribution which is stepped due to the discrete nature of the weighting process.

A further consequence of this stepping process is that at a particular step, recall is higher than precision and then, moving up one step, there is a higher precision becomes higher than recall. This effect is what is observed in the graphs of results for SVMs in Figures 4.6 and 4.7 where best performance was observed for the step where precision is better than recall. Note, here, the number of files is the set of files that all binary classifiers can draw upon to build from whereas the number of features is the number of features per binary classifier.

The graphs indicate that the SVM classifiers are more sensitive to the number of files than multinomial Bayes and indeed, initially, results are poorer for SVMs. However for a high number of files, the SVM classifier performs as well as Bayes. With 4000 files and 2000 features, precision and recall are 0.96 and 0.82 respectively.

These results compare well with those reported by others, even though it has been difficult here to establish break-even points. Joachims [Joachims, 1998], for example, reports a breakeven point using Bayes of 0.72 and for SVM using an RBF kernel the value is 0.864. This at the very least demonstrates that the *Classifiers* software is performing effectively. Indeed the Reuters-21578 corpus may be considered a ‘control’ if further, newer corpora are to be investigated. Such a corpus is the ACM corpus which has not been studied to date and which will be discussed next.

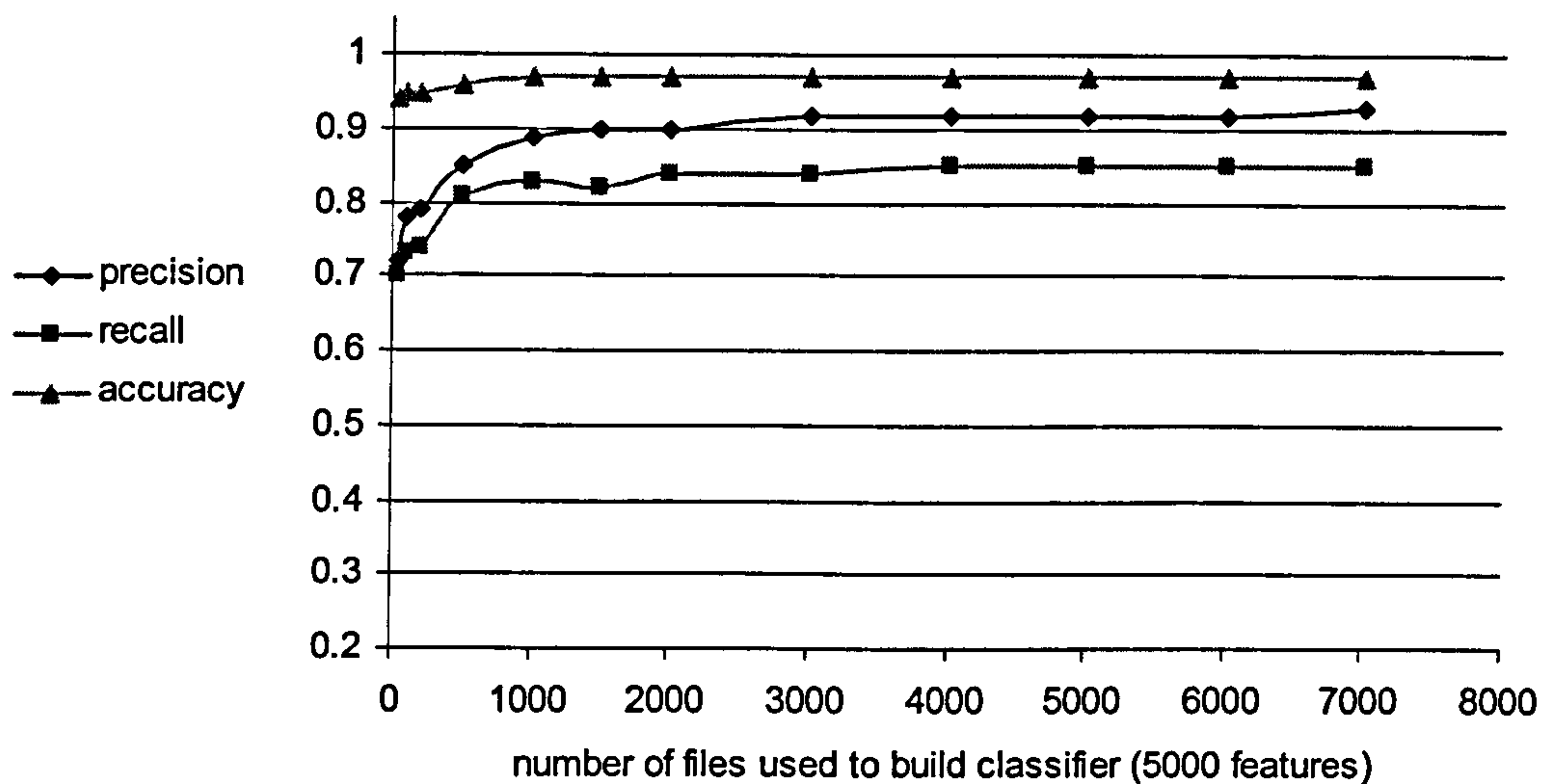


Figure 4.2 Multinomial Naïve Bayes Classifier Performance
Varying Files for Reuters-21578 Classes

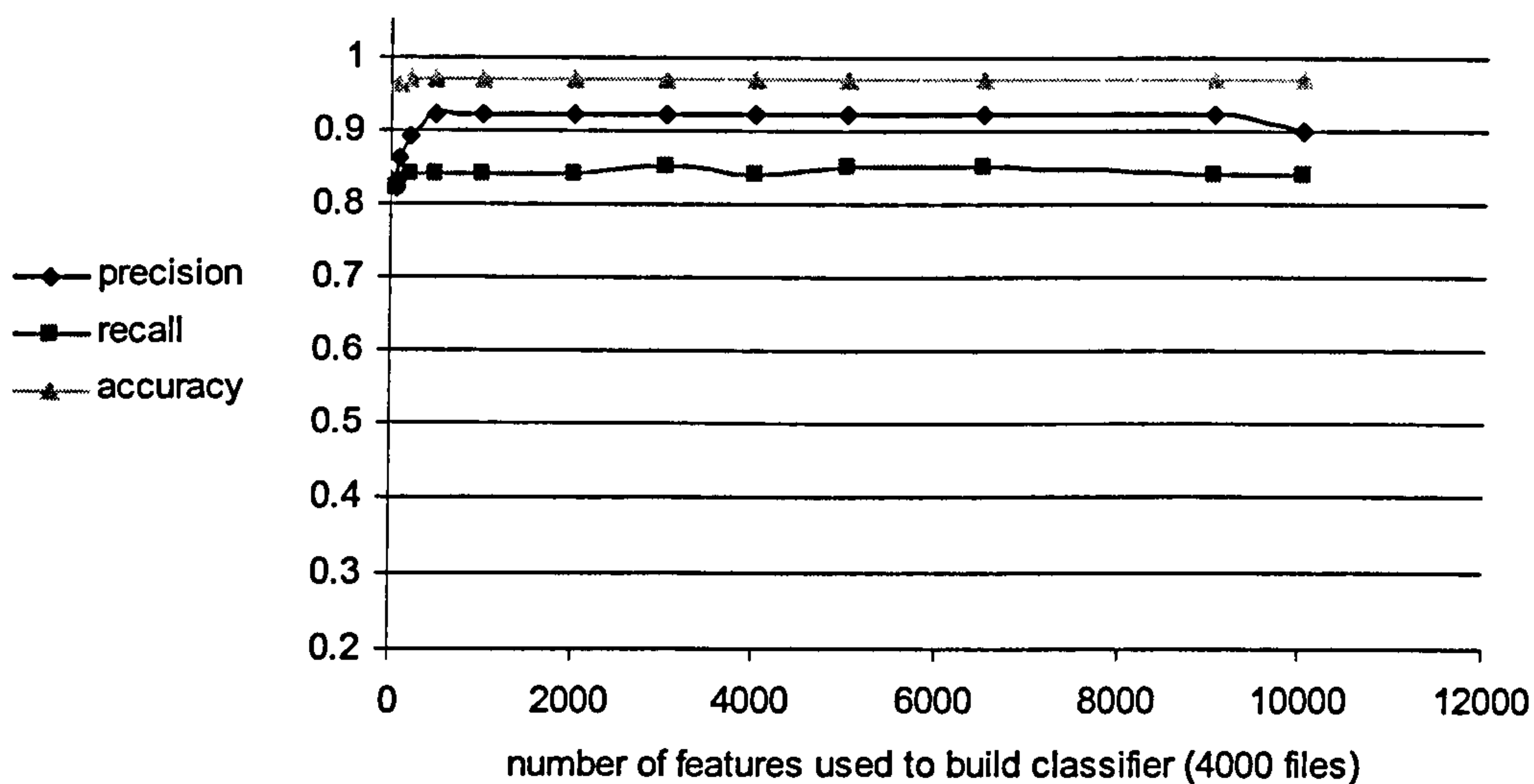


Figure 4.3 Multinomial Naïve Bayes Classifier Performance
Varying Features for Reuters-21578 Classes

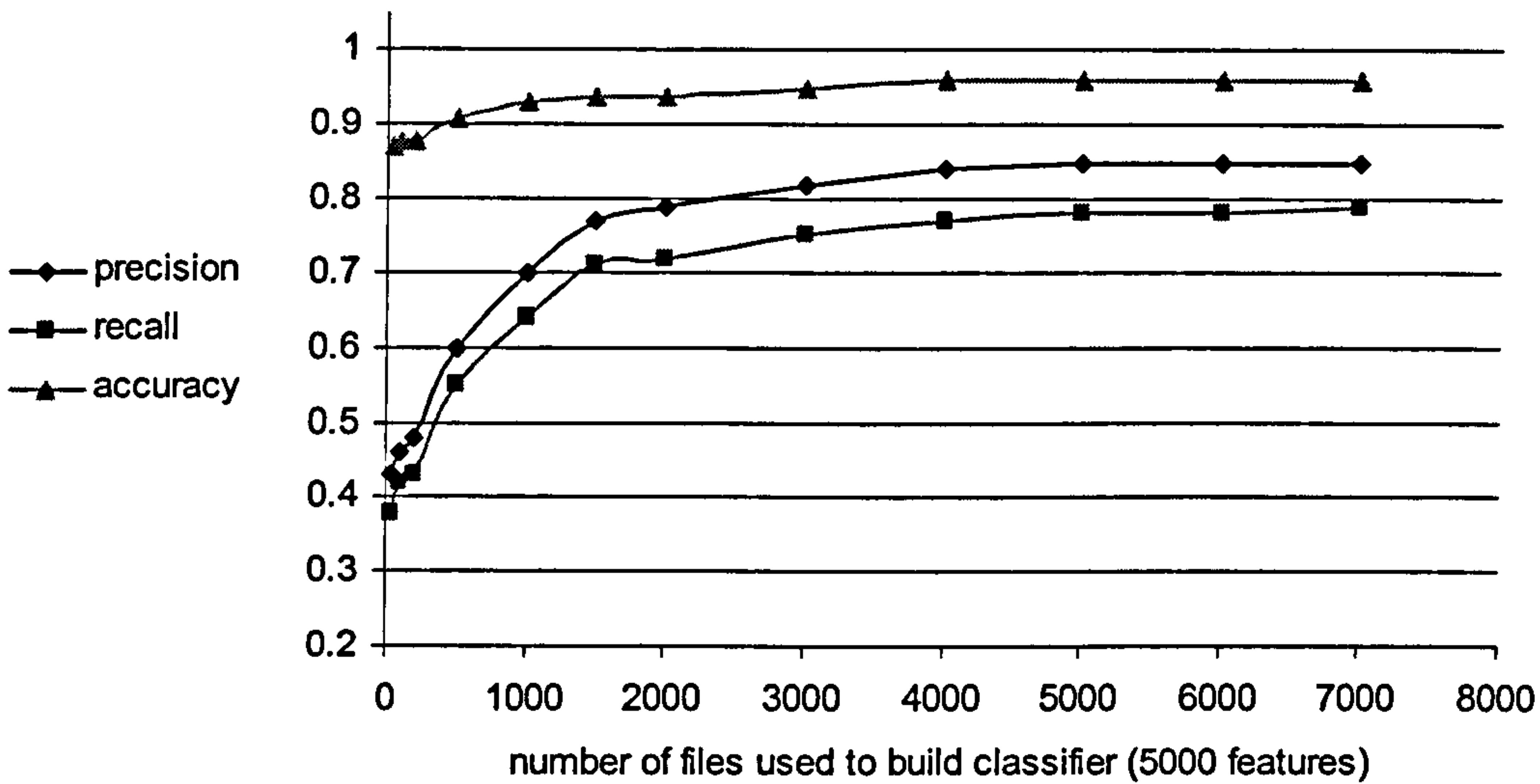


Figure 4.4 Multivariate Naïve Bayes Classifier Performance
Varying Files for Reuters-21578 Classes

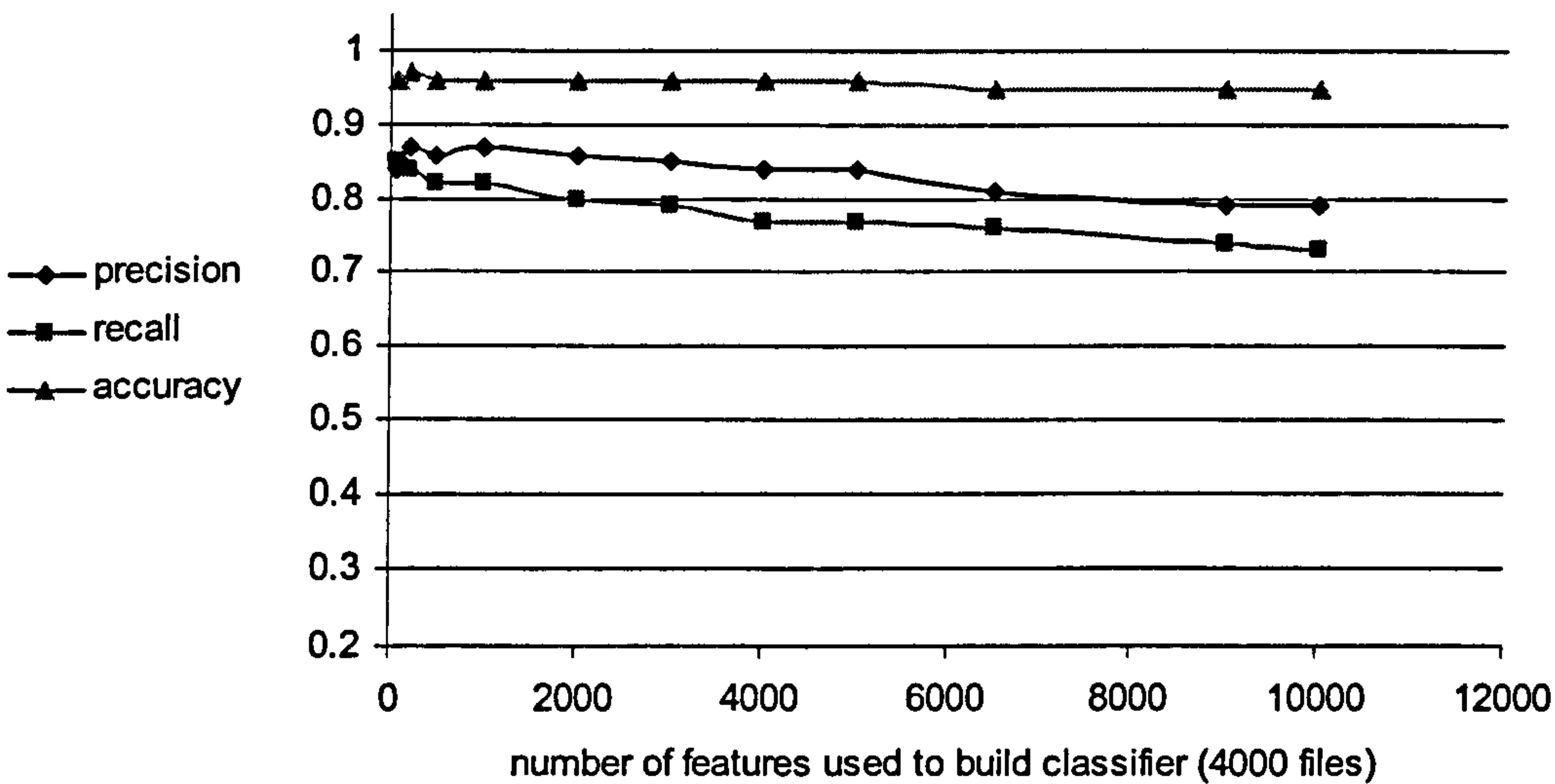


Figure 4.5 Multivariate Naïve Bayes Classifier Performance
Varying Features for Reuters-21578 Classes

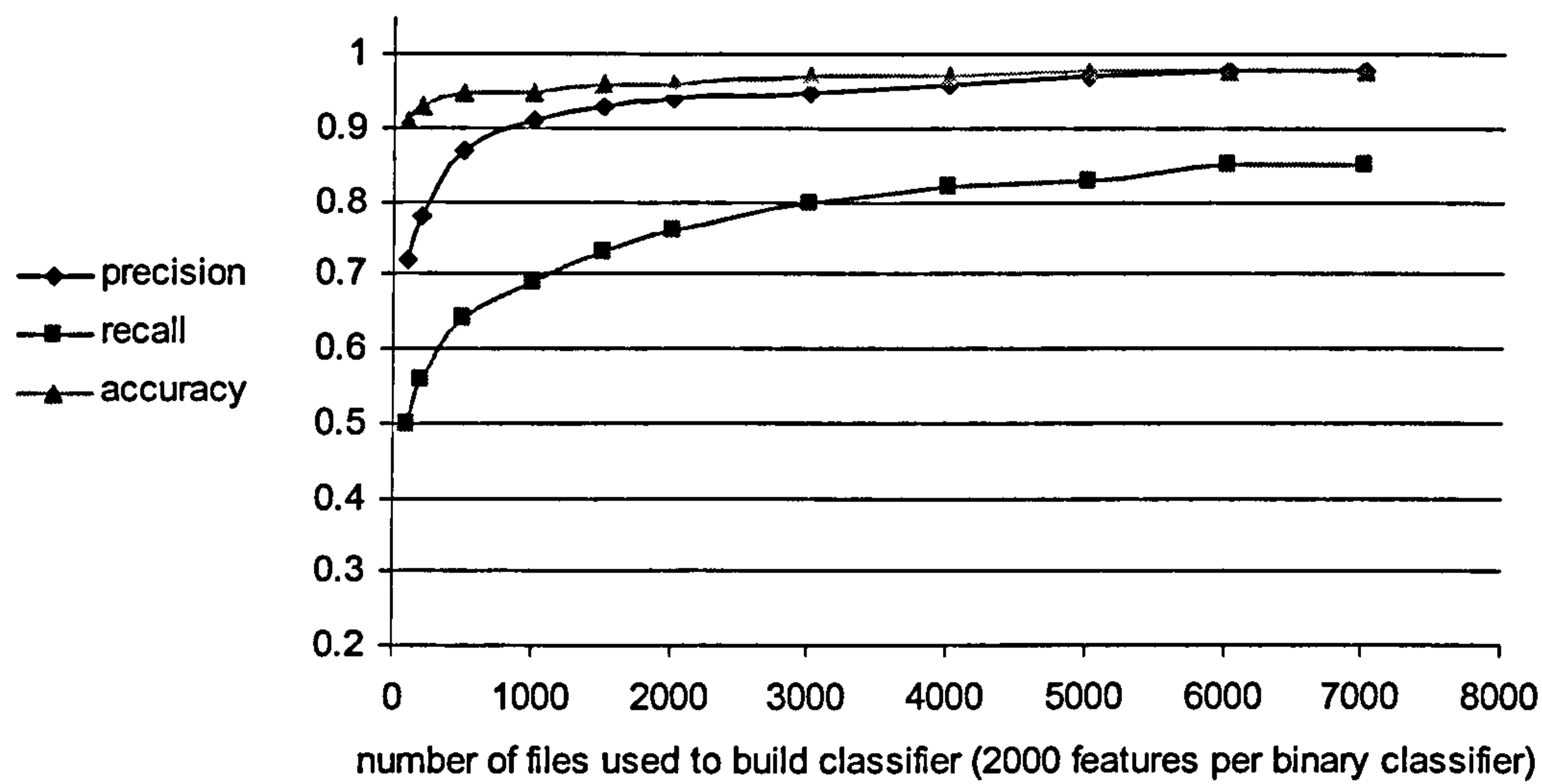


Figure 4.6 SVM Classifier Performance

Varying Files for Reuters-21578 Classes

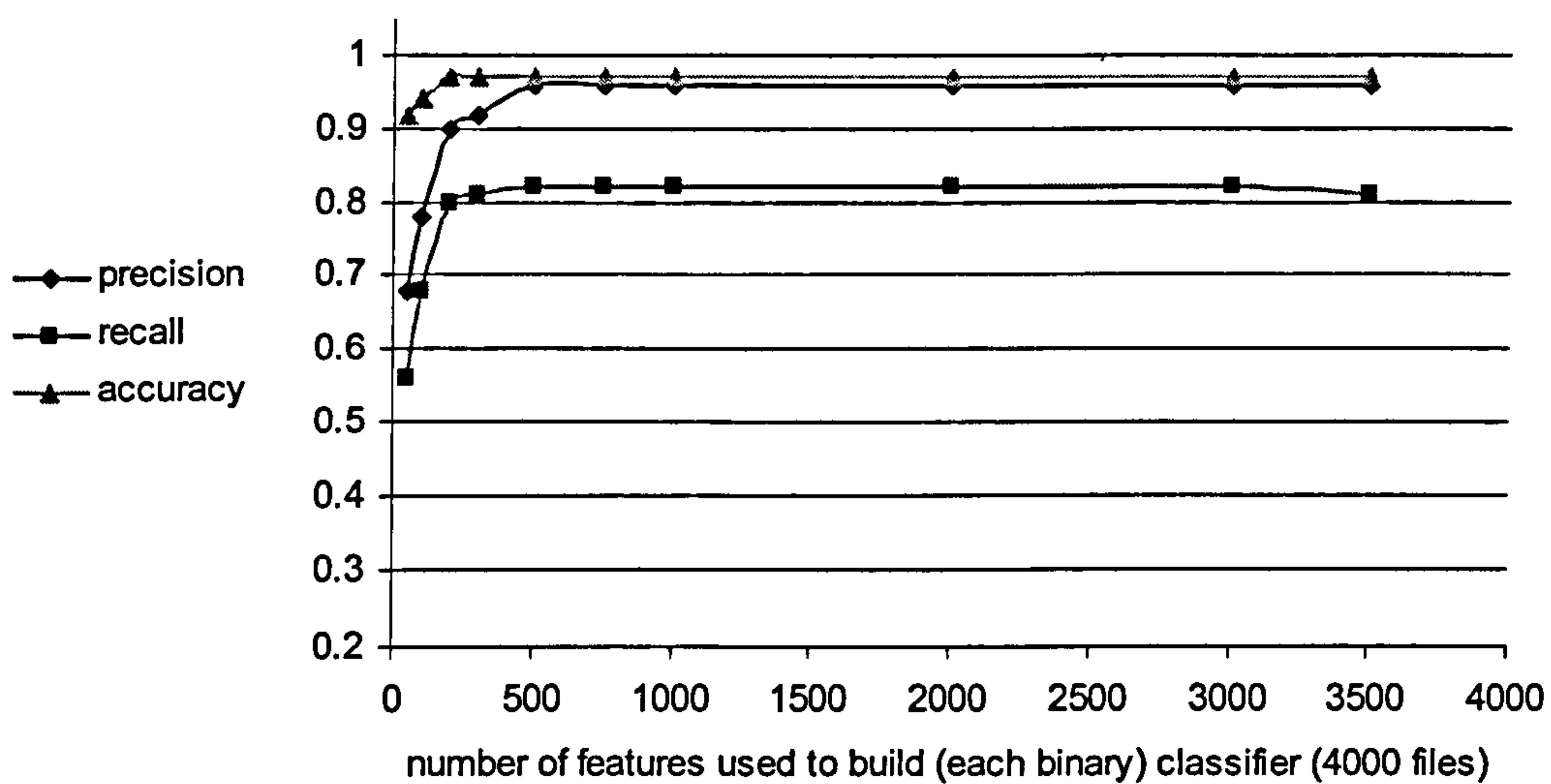


Figure 4.7 SVM Performance

Varying Features for Reuters-21578 Classes

4.5 The ACM Corpus

A presentation of the top two levels of the ACM hierarchy¹ is shown in Figure 4.8. The structure of the scheme is a tree. At the top level, classes A through K are designed to categorize Computer Science documents according to some broad subject areas, for example, ‘Hardware’ or ‘Software’. Subsequent levels further subdivide these areas into topic areas of greater specialization. When a document is manually classified, an individual will typically select the pertinent top level classes and then investigate deeper in the classification scheme to find further pertinent classes. Intuitively, the same approach may be the first approach considered for a machine learning based classifier. That is, we construct a classifier which first awards a document the appropriate classes between A and K and for each of those classes selected, continues to assign classes in the relevant sub-trees, and so on. (Indeed, this was the approach taken in ClassAKT, a hierarchical ACM classifier which will be introduced in Chapter 5).

- A. General Literature
- ...
- B. Hardware
- ...
- C. Computer Systems Organization
 - C.0 GENERAL
 - C.1 PROCESSOR ARCHITECTURES
 - C.2 COMPUTER-COMMUNICATION NETWORKS
 - C.3 SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS
 - C.4 PERFORMANCE OF SYSTEMS
 - C.5 COMPUTER SYSTEM IMPLEMENTATION
 - C.m MISCELLANEOUS
- D. Software
- ...
- E. Data
- ...
- F. Theory of Computation
- ...
- G. Mathematics of Computing
- ...
- H. Information Systems
- ...
- I. Computing Methodologies
- ...
- J. Computer Applications
- ...
- K. Computing Milieux
- ...

Figure 4.8: Extract from the ACM Computer Classification System

¹ Available in full at portal.acm.org

As mentioned previously, no previous effort has been reported to investigate machine learning based classification of the ACM corpus. An effort has been made here to establish a baseline idea of what might be achieved in terms of classifier performance using the Naïve Bayes and SVM classifiers introduced in this work. To do this, classifiers have been constructed and tested to investigate the top level in the hierarchy, classes A through K and one of the next level set of classifications, namely those sub-classes directly below ‘C. Computer Systems Organization’. The ‘C’ sub classes were chosen because they may be considered to be average in terms of text properties (note Table 5.1): there are over 14,000 example documents available over 6 feasible classes. For a class to be considered ‘feasible’, it was only included in the empirical analysis if it contained at least about 50 example documents. In the case of the ‘C’ sub-classes this meant leaving out ‘C.m miscellaneous’. As a side note, this also means at that this stage in the research, further, deeper, levels of the ACM hierarchy would not be considered for empirical analysis because document density at the lower classification points indeed becomes quite sparse.

Figure 4.9 through to Figure 4.14 show some results of investigating the ACM ‘C’ Classes. Figures 4.9 and 4.10 are the results using multinomial Bayes. The first observation is that performance is generally poorer than is observed for Reuters-21578. Here for the ACM ‘C’ classes, good performance comprises a precision of 0.72 and a recall of 0.59. As for Reuters-21578, recall lags precision because of the highly selective nature of the Bayes classifier. Figures 4.11 and 4.12 are similar graphs for multivariate Bayes, where performance is seen to be similar, although not quite as good, as for the multinomial counterpart. Optimal values of precision and recall here are 0.68 and 0.57 respectively. With both strains of the Bayes classifier, performance is observed to be fairly flat after an initial threshold of numbers of files or features is overcome (perhaps 1000 in each case).

Figures 4.13 and Figures 4.14 are graphs depicting the results of employing a SVM classifier on the ACM ‘C’ classes. Here again, similarities may be drawn with the results observed in Reuters-21578. Firstly, after a slow start, the SVM classifier given a large feature set and a large set of files to work with does slightly outperform Bayes with precision and recall reaching optimum values at about 0.62 and 0.82 respectively.

Secondly, a ‘stepping effect’ was again seen in performance due to the classifier being a committee of binary classifiers. In contrast to Reuters-21578, best performance was observed when recall was higher than precision.

Figures 4.15 and 4.16 are some results employing the Naïve Bayes classifier to classify the top level 11 classes between ‘A’ and ‘K’ of the ACM corpus. It is immediately apparent that there is a large difference between precision and recall with recall performing poorly. Optimal values are 0.7 for precision and 0.4 for recall. This means that the classifier is fairly good at pinpointing ‘a’ correct class, but if a document has been attributed to a second or more class, which is often the case in reality, this second class isn’t picked out with the result that recall suffers. The ‘too selective’ nature of the Bayes classifier was discussed for Reuters-21578, but here its effect is so pronounced that the result is a poor classifier.

Results were taken for the ‘A’ to ‘K’ classes employing a multivariate Bayes classifier. Graphs of results have not been reproduced here although, in summary, it was found to perform worse than the multinomial approach with optimal values of precision and recall occurring at 0.62 and 0.34 respectively.

Some results were taken for the ‘A’ to ‘K’ classes employing a SVM classifier. These are shown in Table 4.2 and both points are the average of three classifier builds. Extensive graphs of results have not been shown simply due to the length of time required to make each build - over 6 hours each. Each build requires the generation of a committee of 55 binary classifiers each one constructed from a segment of the ACM corpus consisting of a large number of relatively large documents.

Number of Files	Number of Features	Precision	Recall	Accuracy
6000	8000	0.65	0.50	0.86
10000	8000	0.66	0.53	0.86

Table 4.2 Some Results for ‘A’ to ‘K’ Classes of the ACM Corpus using a SVM

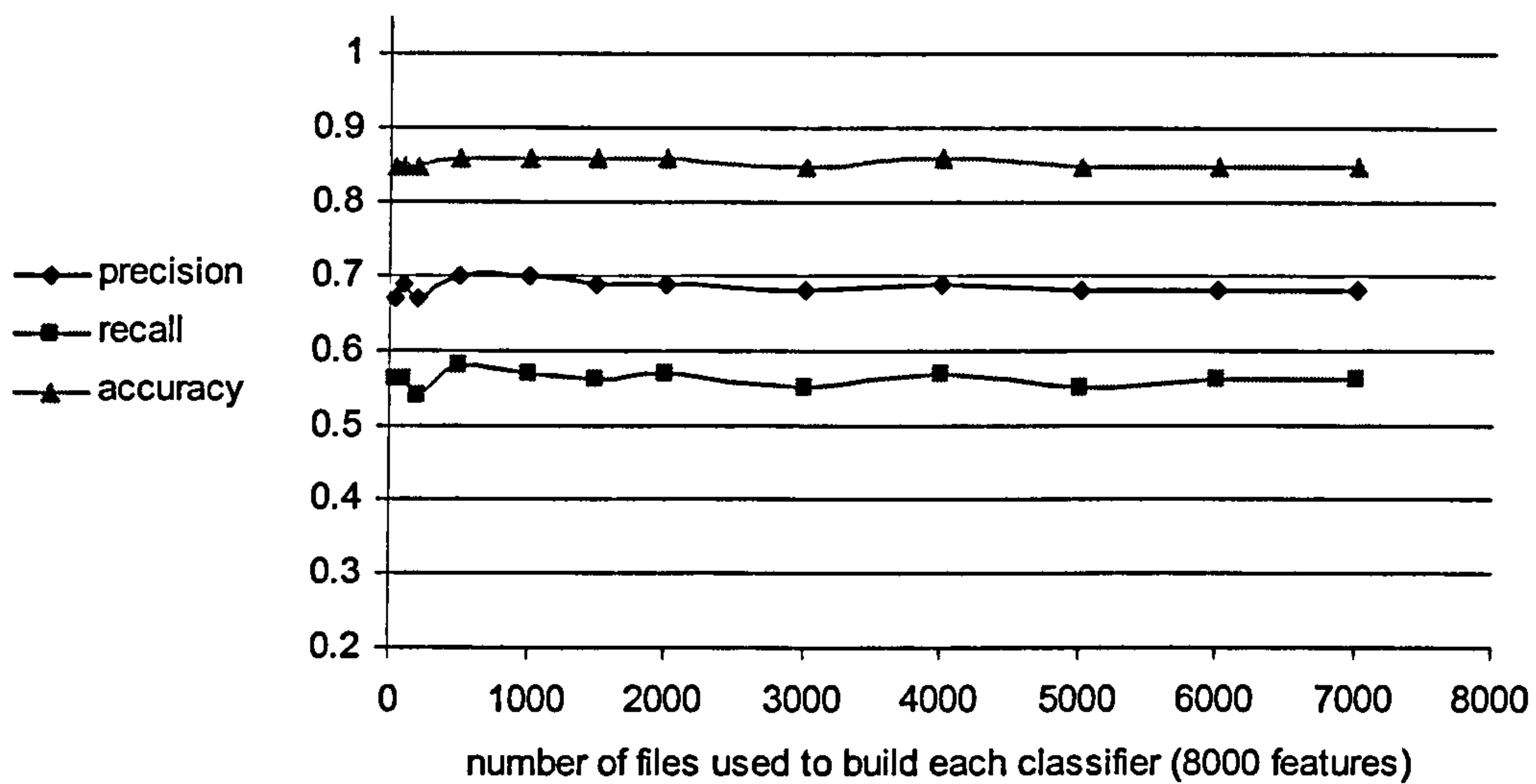


Figure 4.9 Multinomial Naïve Bayes Classifier Performance
Varying Files for ACM ‘C’ Classes

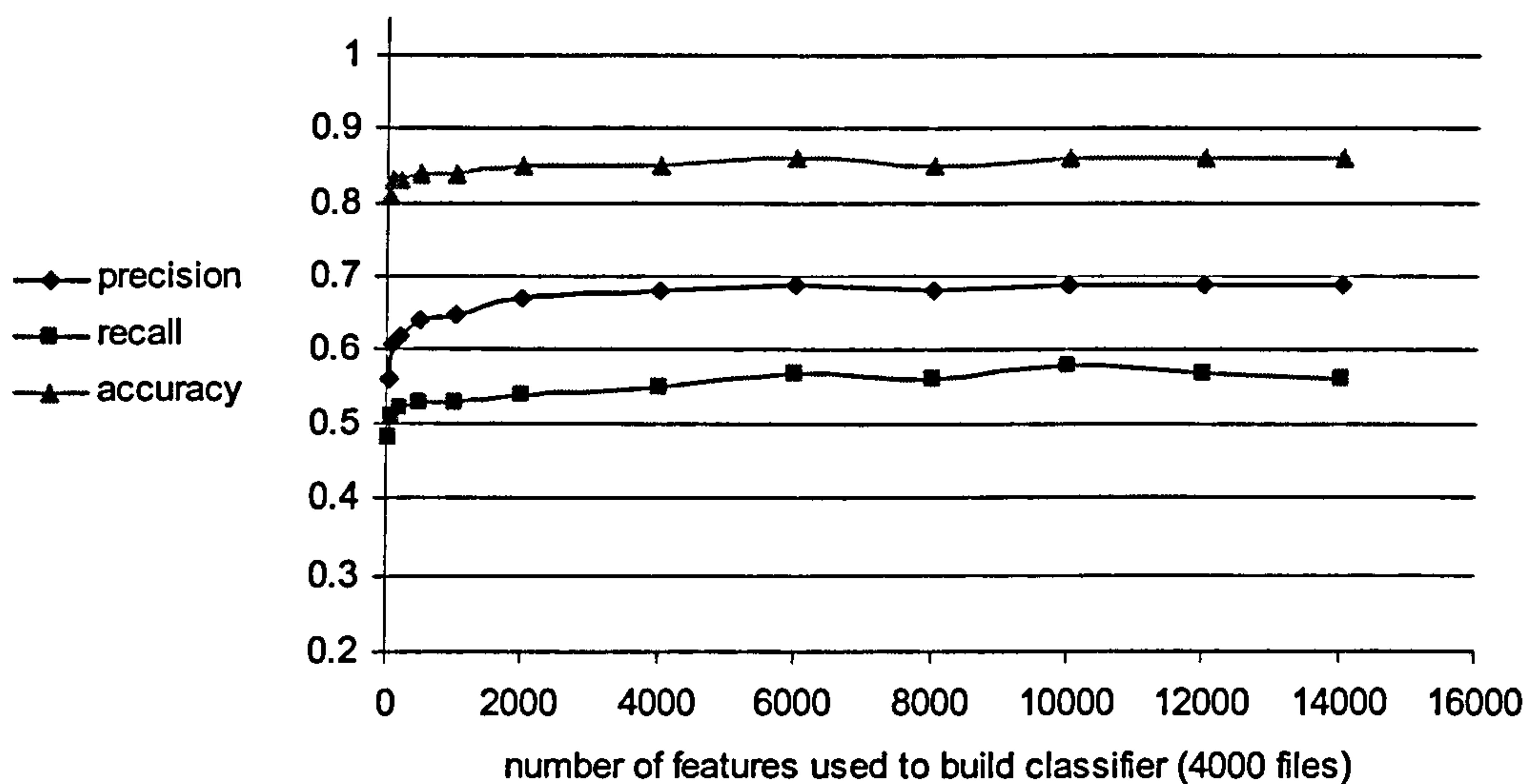


Figure 4.10 Multinomial Naïve Bayes Classifier Performance
Varying Features for ACM ‘C’ Classes

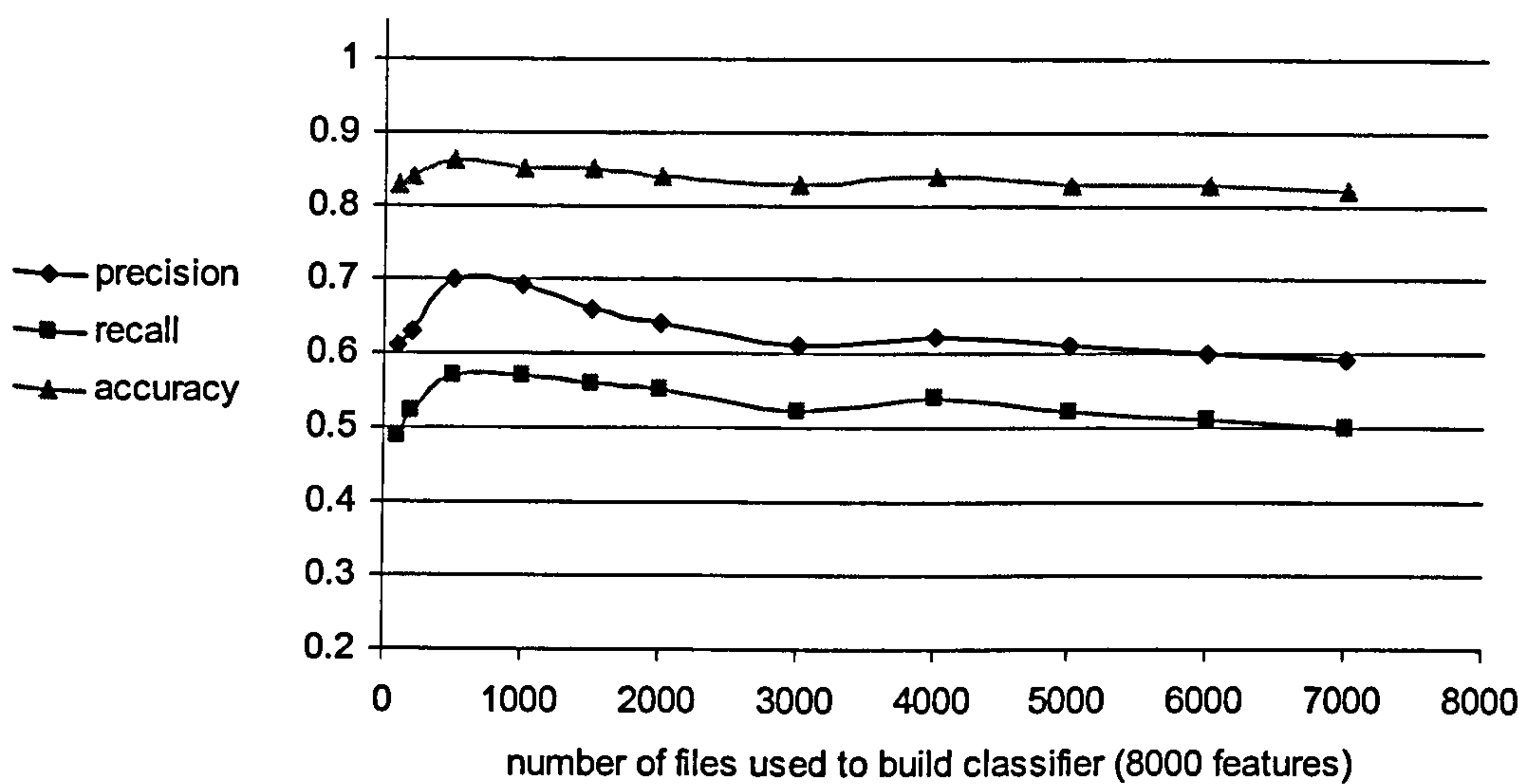


Figure 4.11 Multivariate Naïve Bayes Classifier Performance
Varying Files for ACM 'C' Classes

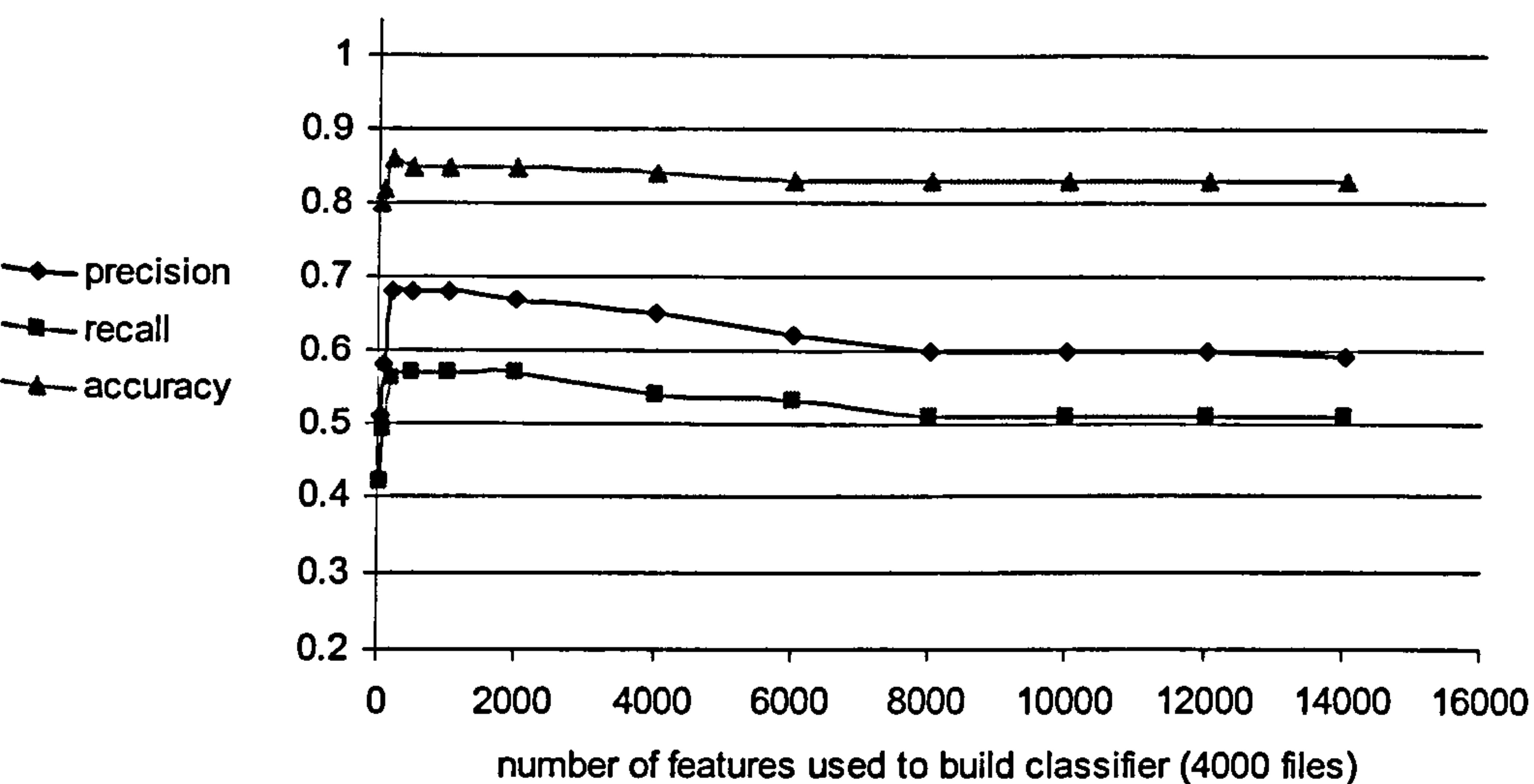


Figure 4.12 Multivariate Naïve Bayes Classifier Performance
Varying Features for ACM 'C' Classes

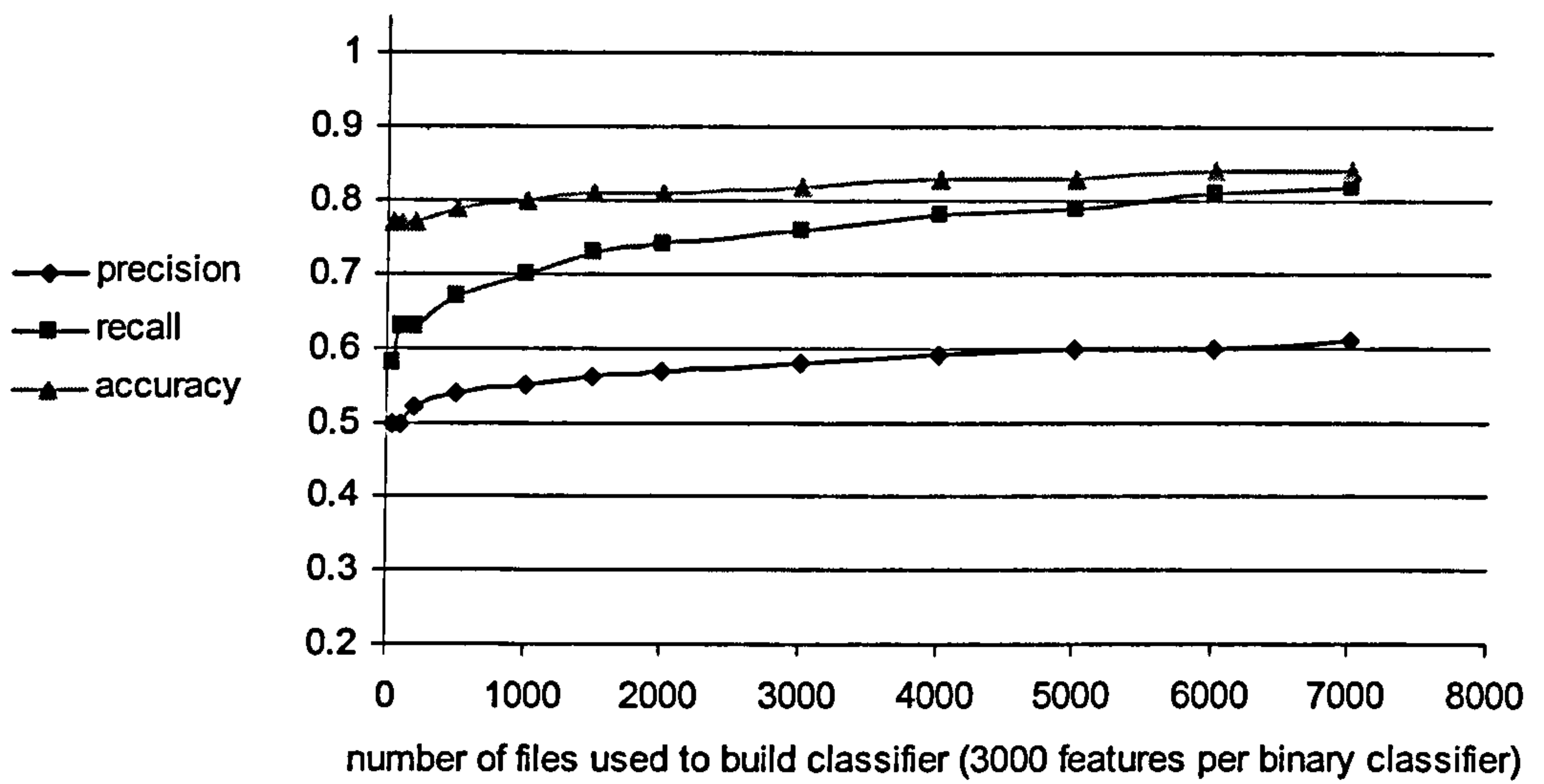


Figure 4.13 SVM Classifier Performance
Varying Files for ACM 'C' Classes

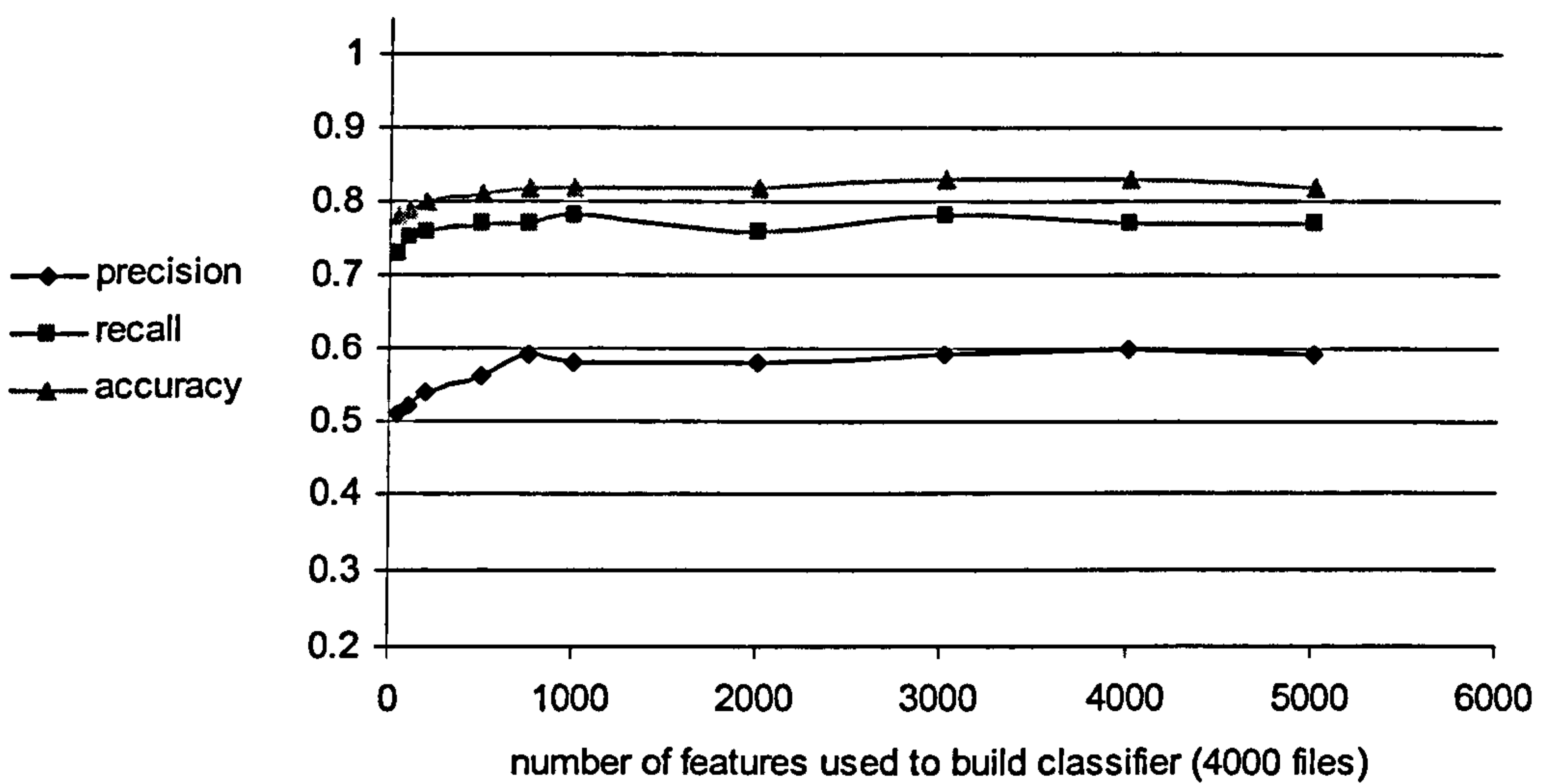


Figure 4.14 SVM Classifier Performance
Varying Features for ACM 'C' Classes

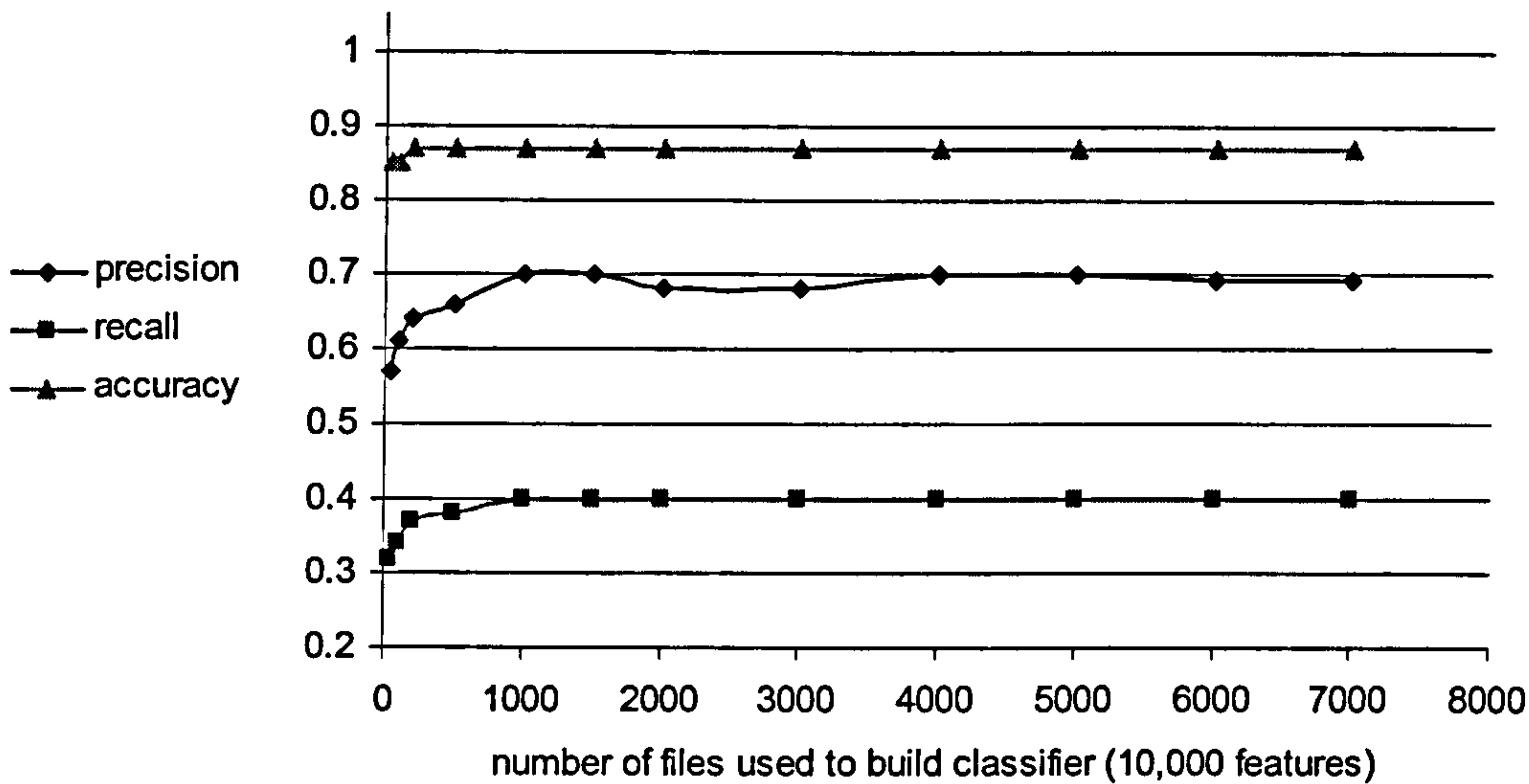


Figure 4.15 Multinomial Naïve Bayes Classifier Performance
Varying Files for ACM ‘A’ to ‘K’ Classes

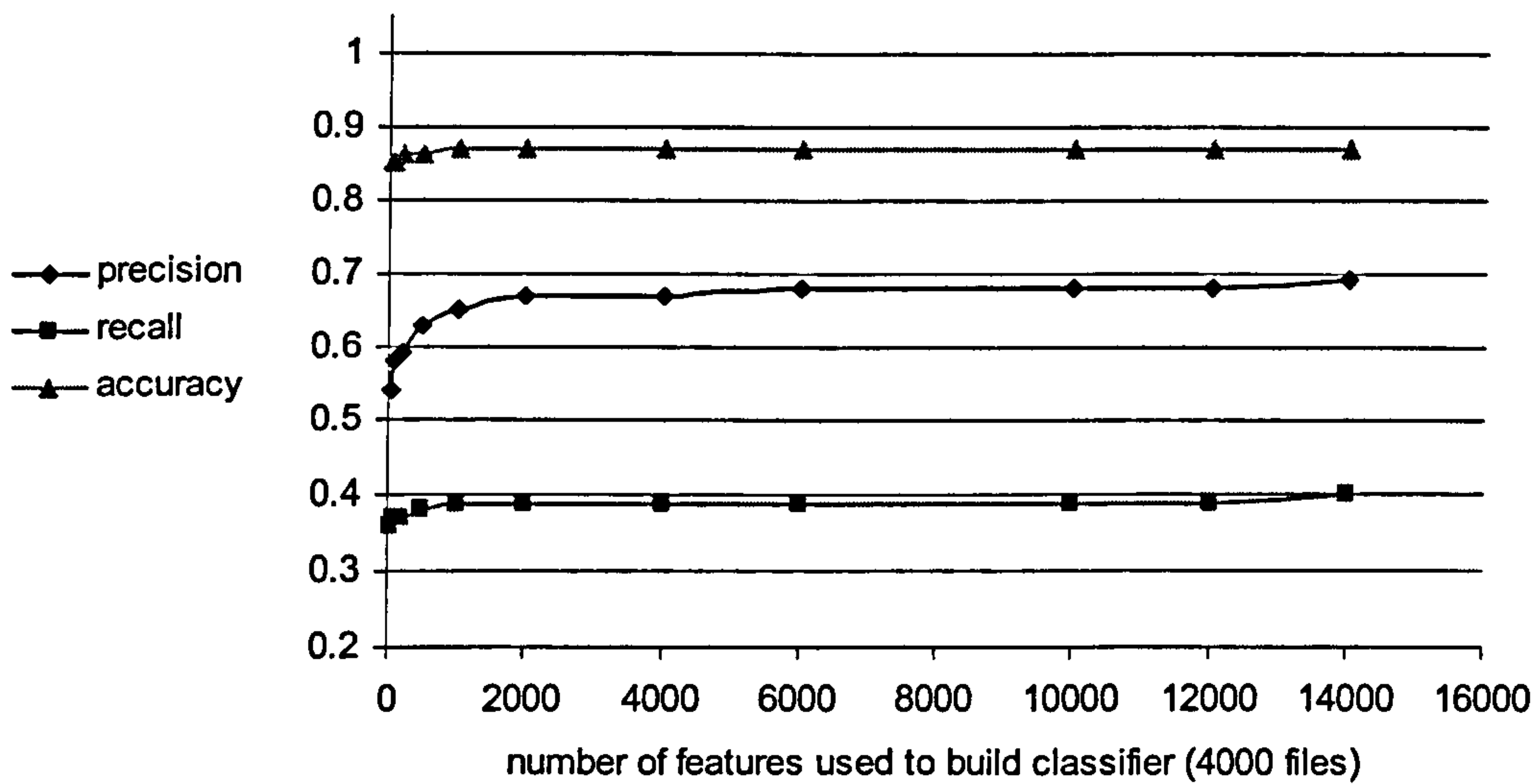


Figure 4.16 Multinomial Naïve Bayes Classifier Performance
Varying Features for ACM ‘A’ to ‘K’ Classes

4.6 Discussion of Results

Broadly speaking, the results published above would indicate that for a corpus with the properties of Reuters-21578 where values for precision and recall are routinely above 0.8, classifiers may be constructed which can actually be trusted to automatically classify unseen documents. For a corpus with the properties of the ACM where the value for recall for the ‘A’ to ‘K’ classes is consistently at a low value of about 0.4, perhaps the best that may be achieved at this stage is a recommender system in which an individual would use an ACM classifier for guidance, but could not rely on the classifier’s decision alone.

Of course, the *Classifiers* framework was developed primarily as a stepping stone to a set of Semantic Web services and the framework itself, while designed to be extensible for further classifier techniques, was intended essentially to provide a baseline idea of classifier effectiveness. As such, there is still wide scope for investigation of alternative classifier algorithms as well as optimization of existing ones and the possibilities of such research are indeed intriguing for a corpus such as ACM. Indeed, it is highly likely that a straightforward alternative approach, perhaps Rocchio or k-NN [Sebastiani, 1999], would provide a better baseline guide to performance than either Naïve Bayes or SVM (which have to date shown themselves to be too selective or too computationally expensive, respectively, for the ACM corpus).

On the other hand, a number of issues arise for the ACM corpus which further highlights the uncertainty in the answer to the question: to what extent is it possible to build a classifier that can be relied upon to automatically classify unseen documents according to the ACM classification scheme?

- Published research on a corpus similar to the ACM is sparse. In particular, textual properties can be very different between different corpora and research on a corpus such as the ACM has been minimal. As mentioned previously, while a Reuters document may contain on average 60 pertinent tokens many of which are highly topic specific, an ACM scientific paper may contain 1,300 tokens in which topic specific tokens, to whatever extent there are any, can be swamped by the large data set.

- Those individuals who are required to submit an ACM classification for a paper they are preparing will invariably complain about the difficulty in selecting the correct ACM categories. It might appear that there is no appropriate category for their document or that there are too many. The ACM hierarchy tree can appear difficult to navigate so that an appropriate class deep in the tree hierarchy is hard to find. Also, it is questionable if the ACM classification scheme is fit for purpose. Is it too arbitrary in nature, too general or perhaps it has become outdated. The net result of this is that if we give two researchers the same paper to classify and it is quite likely they will come up with different classifications. This has the effect of smudging the boundaries between classes thereby making automatic classification more difficult.
- Classifications in the ACM hierarchy tend to be very broad by their nature and there can be considered to be a degree of overlap in the subject content of their categorizations. For instance, it would perhaps be unsurprising if ‘H. Information Systems’ and ‘J. Computer Applications’ were found to share a fairly similar set of keywords in their feature sets.

Clearly more research on the ACM and similar corpora would be valuable. Perhaps a more complete analysis could establish a set of heuristics for a corpus which make it amenable for the construction of trustworthy automatic classifiers. Given this set of heuristics, it might be found that the ACM corpus could be said to fall outside their bounds and that a successful automatic classifier based on this corpus which can replace a human is indeed not possible. Such a conclusion might lead to the establishment of classification schemes which are designed with more appropriate care and attention shown towards an eventual goal of creating effective automatic classifiers.

4.7 Summary

This chapter has introduced a software framework which permits the construction and testing of various machine learning approaches to text classification, and in particular it has investigated the effectiveness of two particular technologies, Naïve Bayes and Support Vector Machines, on two corpora, Reuters-21578 and the ACM corpus of Computer Science related documents. Reuters-21578 is a much studied corpus and results

indicate that it is amenable to automatic classification. The ACM corpus and classification scheme is new to study as a subject of machine learning techniques. It is an example of a popular established system responsible for managing large, in this case scientific, documents and in particular it is an example of a widely adopted current classification scheme for which it would be most useful if full automatic classification could be enabled.

It is argued that further research is required on various corpora in an effort to establish heuristics to describe what specification of corpus is amenable to automatic classification. In a mature Semantic Web, the ability to take an arbitrary corpus and utilize a set of Web services to create a useful classifier according to some arbitrary classification scheme would indeed be a highly useful activity for which the work in this thesis may be considered a primary step. To succeed in this goal, as the work here has shown, further investigation into textual properties of corpora and the discriminatory value of classes in a classification scheme need to be investigated much more fully. In addition, more basic guidelines such as the number of files and features required to create effective classifiers need to be established. This has been the motivation for the way in which results have been published in this chapter: the graphs indicate at what ranges of files and feature numbers classifier performance is most effective for a given machine learning technology and is an primary step towards a set of heuristics.

The following chapters deal with making this classifier technology a real application on the Semantic Web. In particular, suitable ontologies are detailed to describe the domain in which the classifiers operate and a Semantic Web service framework is designed and built (based on the *Classifiers* software) on top of these ontologies. Before this, however, the next chapter introduces ClassAKT, a ‘hardwired’ Semantic Web service which classifies documents according to the ACM classification scheme.

Chapter 5 ClassAKT: A Semantic Web Service

5.1 Introduction

As discussed in the introduction, a key motivation for the work undertaken in this thesis is to investigate and demonstrate the use of services which enable the automatic generation of metadata for the emergent Semantic Web. This chapter details such an effort in the form of an actual Semantic Web service¹, ClassAKT, which makes use of machine learning techniques to instantiate a service which classifies scientific documents related to computing according to the top two levels of the ACM classification hierarchy. This chapter details how this Web service was created, its OWL-S interface and some preliminary results indicating the classifier's performance in classifying documents.

5.2 Classifier Construction

Chapter 3 gave a detailed introduction into the requirements necessary to construct a text classifier based on machine learning techniques. Essentially, the requirement for this is a tagged corpus of documents, that is, the set of documents and the classes associated with each of them according to some classification scheme. The corpus selected for primary study in this thesis is the one made accessible by the Association for Computing Machinery (ACM) at their portal at portal.acm.org. This corpus shows a representative cross section of scientific documents pertaining to a wide variety of subject areas within the field of Computer Science. In addition, documents maintained by the ACM are each

¹ Available at <http://classakt.aktors.org/classifier/>

associated with one or more classifications taken from the ACM's own classification scheme¹ (discussed previously in Chapter 4).

The overall process required to make a classification service available on the Web is illustrated in Figure 5.1. Here, the boxes indicate the steps necessary for the creation of the Web service itself.

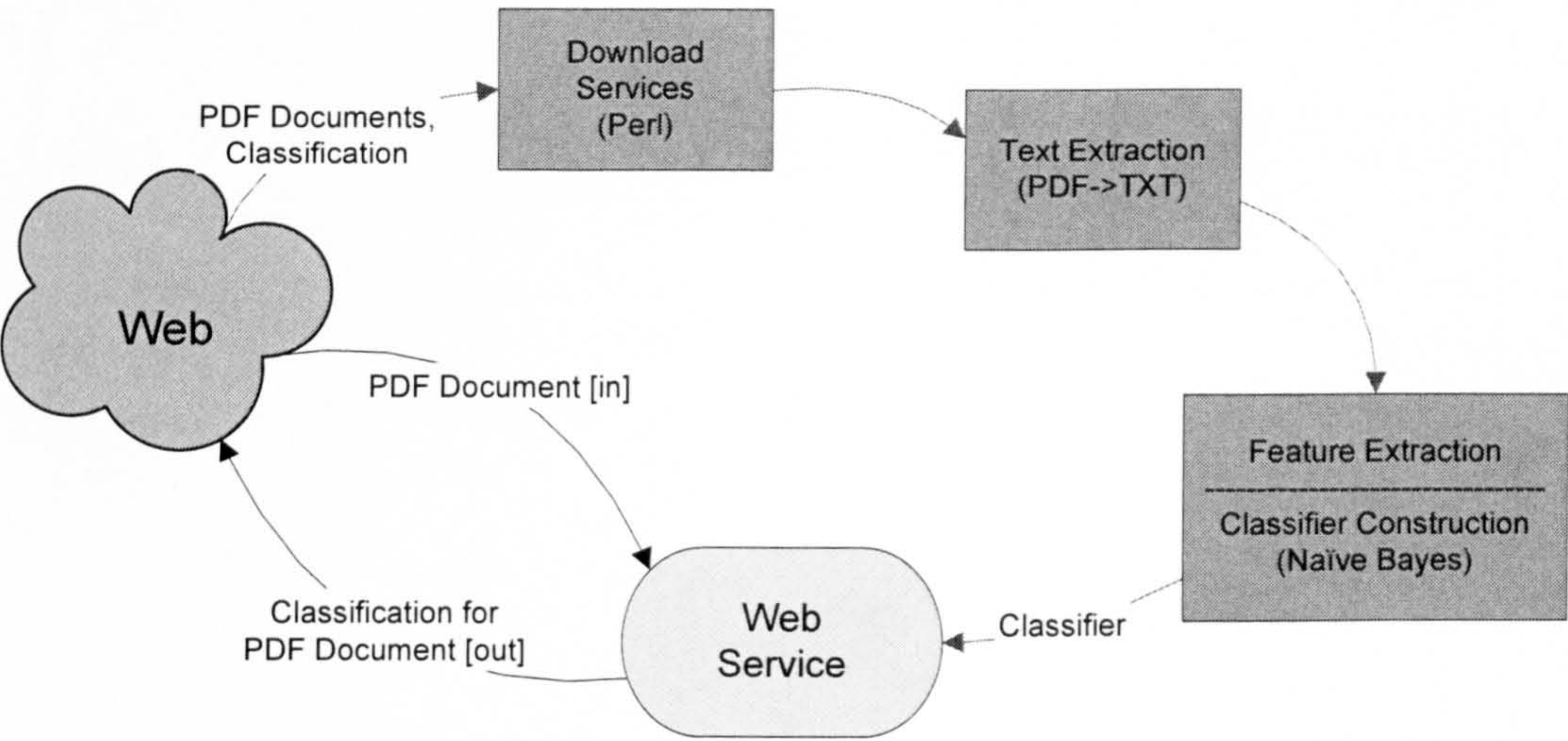


Figure 5.1: System Diagram for ClassAKT

Firstly, access is required to the ACM corpus. In a mature Semantic Web, it would be expected that such information would be available as RDF metadata which would map to some ontology and access to the data (in this case scientific documents and their metadata) would be facilitated by generic Semantic Web tools. While this exemplifies the shortcomings of today's Web and provides a use case for why the Semantic Web will be useful, it does not help with the task at hand of obtaining data easily from today's Web. Today the simplest way of accessing such data is via screen scraping techniques whereby some tool or programming language is utilised to access the required information by processing HTML currently available on the Web.

¹ Available at <http://www.acm.org/class/1998/>

In practice, scripts in the programming language Perl¹ were employed to download documents from the ACM portal together with their associated metadata (in particular, classifications). In total, over 119,000 PDF documents were downloaded. Table 5.1 gives an idea of the distribution of these files across the 11 top level classes of the ACM hierarchy. It should be pointed out, however, that since the ACM classification is a multi-class system, a single document may be attributed to certainly one but quite possibly two or more classes. The way these figures were collated means that the same document may well appear under two or more classes in the counts indicated.

Class	Number of PDF Files	Number of Sub-classes
A. General Literature	1,657	4(3)
B. Hardware	8,912	10(8)
C. Computer Systems Organization	14,113	7(6)
D. Software	20,518	6(4)
E. Data	2,172	7(5)
F. Theory of Computing	1,339	6(4)
G. Mathematics of Computation	5,815	6(4)
H. Information Systems	19,671	7(5)
I. Computing Methodologies	20,794	9(7)
J. Computer Applications	9,556	9(7)
K. Computing Milieux	14,605	10(8)

Table 5.1: Class Distribution of ACM Documents

Also indicated is the number of sub-classes beneath each of the top level classes and the figure in brackets for each of these indicates the number of these sub-classes for which there is a sensible number of documents for that class so that machine learning based classification remains a reasonable proposition. Although the ACM classification scheme extends further than these two levels, document density becomes quite sparse and classification untenable with the techniques employed.

¹ See www.perl.org

Raw text was next extracted from the PDF files. This was achieved with a Java library, `jpedal.jar`, until fairly recently freely available from the developers¹. To construct ClassAKT, a well known library, Weka was employed. This library² is a Java library which is a collection of machine learning algorithms for data mining tasks. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. In particular, ClassAKT uses Weka's Naïve Bayes algorithm (the Java 'NaiveBayes' class) which provides a variation on the Bayes algorithm in which variables are treated as continuous and are modeled on a Gaussian distribution, see [John & Langley, 1995]. Features were generated by information gain as detailed in Chapter 3.

Firstly, a program was written to evaluate the performance of the Weka algorithm. As an example, for the direct sub-classes of 'C. Computer Systems Organization', that is the 6 classes C.0 through to C.5, it was found that incorporating 2000 plus features and 4000 plus documents to build a classifier led to fairly stable values of precision and recall, namely 0.55 and 0.6 respectively.

Then, making use of these results, a Semantic Web classifier service was created as detailed in the following section.

5.3 A Semantic Web Service

The Web service is available at <http://classakt.aktors.org/classifier> and is implemented using the Java 2 Enterprise Edition (J2EE) framework. It is configured to classify PDF or text documents according to the top two levels of the ACM classification hierarchy. So, for example, if it was asked to classify a document titled 'VLSI layout of trees into grids of minimum width', it would be reasonable to expect, as part of the predicted classification, to label the document under 'B. Hardware' as a top level class and 'B.7 Integrated Circuits' below that. (A document may be matched against multiple classes at a given level in the hierarchy). An HTML form allows a document to be classified and the results returned in a user friendly Web page. Alternatively, results may be returned in an XML format.

¹ At www.jpedal.org

² Available from www.cs.waikato.ac.nz/~ml/weka/index.html

An essential component of the Semantic Web is a rich and capable framework which makes Web services available and describes them in terms of current best practices in knowledge representation. An OWL-S description of ClassAKT is listed in Appendix A¹. The service is defined in terms of the following components:

- **ClassAKTService.owl**: Reference point for the service which describes links to the other OWL-S components.
- **ClassAKTProfile.owl**: Performs the role of specifying registry information, specifies what the service does. ClassAKT declares in terms of the appropriate ontologies that the service accepts a url as an input parameter and returns a set of ACM classes.
- **ClassAKTProcess.owl**: Controls interaction with the service. A specification of ClassAKT as a simple atomic process in which a link is made in a process ontology between the single URL input and the output consisting of document classifications.
- **ClassAKTGrounding.owl**: Maps OWL-S inputs/outputs to WSDL message parts.
- **ClassAKTGrounding.wsdl**: WSDL grounding.

Inputs and outputs are defined in terms of the AKT reference ontology. This now makes it possible to meet one of the objectives outlined in the introduction, namely by being able to programmatically access the service via its Semantic Web based OWL-S interface, to enable automatic population of classification fields for computer science documents within the AKT Triplestore.

5.4 Summary

This chapter has introduced a Web service, ClassAKT which classifies computing related documents according to the ACM classification scheme. Efforts to provide working services for the emergent Semantic Web are necessary since only through practical implementation can limitations in the current technology and requirements for a

¹ At time of writing, see also <http://classakt.aktors.org/services/owl-s/0.9/ClassAKTService.owl>

future mature framework be established. Services that generate metadata are particularly useful because the wealth of available documents on the Web makes manually generating such information infeasible and yet it is a key requirement for enabling search and navigation.

As far as limitations are concerned, the chief limitation at the time of implementation was a distinct lack of available tools for authoring and executing OWL-S scripts. Two key requirements may be pinpointed which will lead to a more mature framework and will indeed be addressed in the remainder of this work. Firstly, a lack of appropriate ontologies and a realisation that the creation of ontologies must be carried out with a view to maximising their generic usability. If services are to be useful then their reusability must be maximised. For example, a classifier service might be considered less valuable if its inputs and outputs map to an obscure ontology whereas if a more general ontology is used, based on a more abstract technology domain (and later in this work, a digital archive will be considered such a domain), then reusability is likely to be enhanced. Secondly, while ClassAKT may be considered useful, a far more useful and powerful service is one that might be considered for generating such services on the Semantic Web i.e. a ‘Web service generator’. That is, given a service, a feature set, a corpus and a classifier configuration we then ask such a Web service generator to produce a custom-built version of ClassAKT.

Chapter 6 Document Archives

6.1 Introduction

A Web service on the Semantic Web is required to be grounded in one or more domains of discourse or ontologies. For a text classification Web service it was argued that the most appropriate such domain is that of document archives. This chapter introduces the areas in which workers in the area of document archives have made important contributions and goes on to highlight some key standards that the design of such an archive on the Semantic Web should reflect.

6.2 The Development of Digital Archives

The requirement for, and development of, traditional archives has been a topic of investigation for almost as long as the written word has existed. These archives may be further subdivided according to the three distinct types of information institution which hold the documents.

- **Libraries:** Identify, acquire, preserve and provide access to the world's published knowledge.
- **Museums:** Identify, acquire, preserve and exhibit unique, collectible or representative objects.

- **Archives:** Identify, appraise, preserve and make available documentary materials of long-term value (essential evidence) to the organisation, or public, that the archives serve.

Each of these types of institution has seen the benefits offered by computer technology in the twentieth century. A major research and development effort has been undertaken to exploit the opportunities offered by the digital information environment.

However, the established archival community has been well positioned to offer its own experience in archive management, as new technologies come online. In her report published in 2000, Gilliland-Swetland [Gilliland-Swetland, 2000] outlined the following five areas in which this community has, and will continue, to offer a valuable contribution to the development of digital archival systems.

- **Integrity of Information:** This concern has two aspects: checking and certifying data integrity (associated with technical processes such as integrity checking, certification, digital watermarking, steganography, and user and authentication protocols) and identifying the intellectual qualities of documents that make it authentic (associated with legal, cultural and philosophical concepts such as trustworthiness and completeness).
- **Metadata:** Additional data that describes information objects in an archive. For example, contextual description, indications of relationships between collections of materials, annotations that have accrued around information objects, documentation of intellectual property rights and documentation of processes that the information objects have undergone, such as reformatting and migration.
- **Knowledge Management:** Reference to the practices, skills and technologies associated with creating, organizing, storing, presenting, retrieving, using, preserving, disposing of and re-using information resources to help identify, capture and produce knowledge.
- **Risk Management:** Assessment of the risk in terms of ensuring the reliability, authenticity, appropriate elimination and preservation of digital information. Risk

management may be seen as a requirement to fulfil the demands of external agents such as auditing, quality control, insurance and law.

- **Knowledge Preservation:** More than simply preserving media or content. Knowledge Preservation involves preserving the intellectual integrity of information objects, including capturing information about the various contexts within which information is created, organised and used; organic relationships within other information objects; and characteristics that provide meaning and evidential value. A major study in 1994 initiated by the Commission on Preservation and Access and RLG [CPA, 1996] set out to investigate knowledge preservation and sets the foundation for many projects undertaken ever since. It emphasized the term ‘data migration’ for maintaining credible and consistent knowledge sources over time and technology change.

The rest of this chapter focuses on some key current efforts which aim to set standards that attempt to satisfy some of the five issues outlined above. Although issues pertaining to document archives predate the Internet, it is the advent of the Internet which is of particular interest in this work. The Internet in itself has brought about an enormous increase in the amount of knowledge available to the computer user. However, it can be argued that the five issues, for the wealth of data available on the Internet, have yet to be dealt with effectively or even approached in any significant way whatsoever. The Semantic Web initiative, in its relation to document archives, has the opportunity to address this shortcoming. By formalising the way knowledge is represented, it offers for the first time a framework on top of which standards may be created to initiate a truly global network of machine processable knowledge resources, of which archives are one of the most important examples.

6.3 Open Archival Information System (OAIS)

The Reference Model for an Open Archival Information System is a product of the Consultative Committee for Space Data Systems (CCSDS) and has been recognised as an ISO standard. According to this standard [CCSDS, 2002], *‘An OAIS is an archive, consisting of an organization of people and systems, that has accepted the responsibility*

to preserve information and make it available for a Designated Community'. The model emphasizes long-term digital information preservation and access within a framework that fosters understanding and increased awareness of archival concepts. It facilitates the description and comparison of various architectures of archives which may be developed without actually prescribing any specific implementation details.

Emphasis on the long term preservation of information is reflected in the model of an OAIS Information Object shown in Figure 6.1. Here a Data Object may be an image file or a document formatted to a specification such as PDF. However, over time or context, interpretation of this data may become impossible if the surrounding knowledge base fails to provide sufficient detail such as the specification of the image or document or even the language such as English in which the written word is presented. This additional detail is referred to in an OAIS as Representation Information and the Data Object plus its Representation Object yields an Information Object.

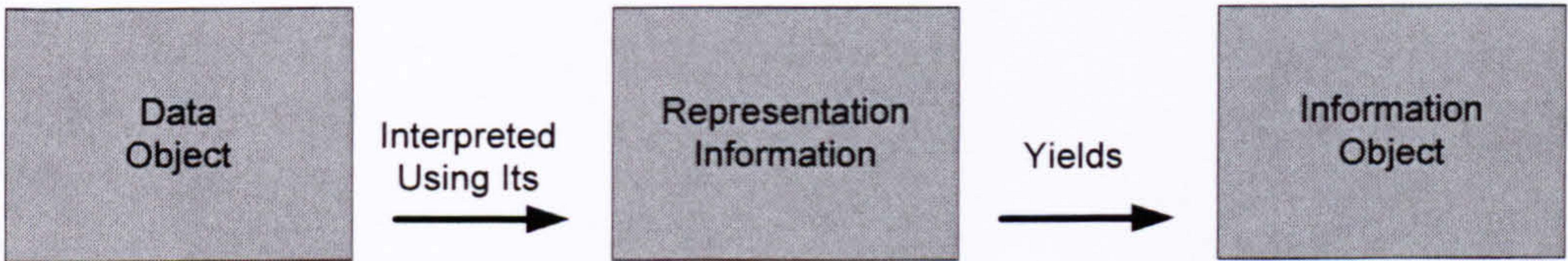


Figure 6.1: Generation of an Information Object

Figure 6.2 is the functional model of an OAIS. An OAIS archive is acted upon by three external agents, namely **Producers** (persons or client systems which provide information to be preserved), **Consumers** (persons or client systems that access an archive to find and acquire preserved information of interest) and **Management** (the role played by those who set overall OAIS policy as one component in a broader policy domain). Within an OAIS there are six functional entities. The diagram shows these entities together with the information flow between them indicated by the lines connecting them.

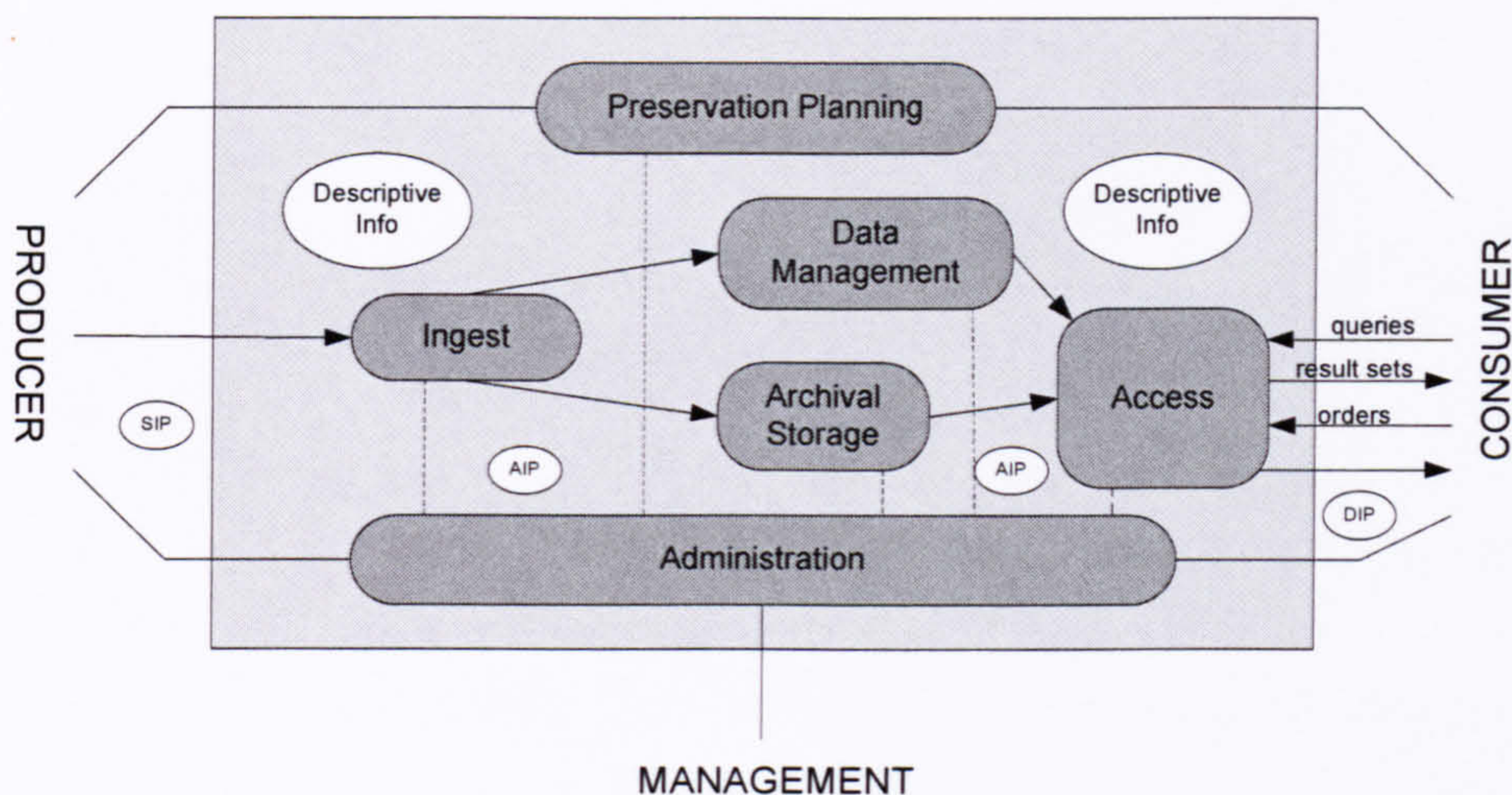


Figure 6.2: OAIS Functional Entities

Information passed into, out of and within an OAIS takes place involving discrete transmissions of what is known as **Information Packages**. An Information Package is shown schematically in Figure 6.3. It comprises **Content Information** (the original target of preservation together with associated Representation Information) and **Preservation Description Information** (PDI) (needed to preserve the Content Information with preserving information specified as Provenance, Context, Reference and Fixity) wrapped in **Packaging Information** which either actually or logically, binds, identifies and relates the Content Information and PDI. The **Descriptive Information** is that information which is used to discover which package has the content information of interest. There are three variants of Information Packages namely the **Submission Information Package** (SIP), **Archival Information Package** (AIP) and **Dissemination Information Package** (DIP).

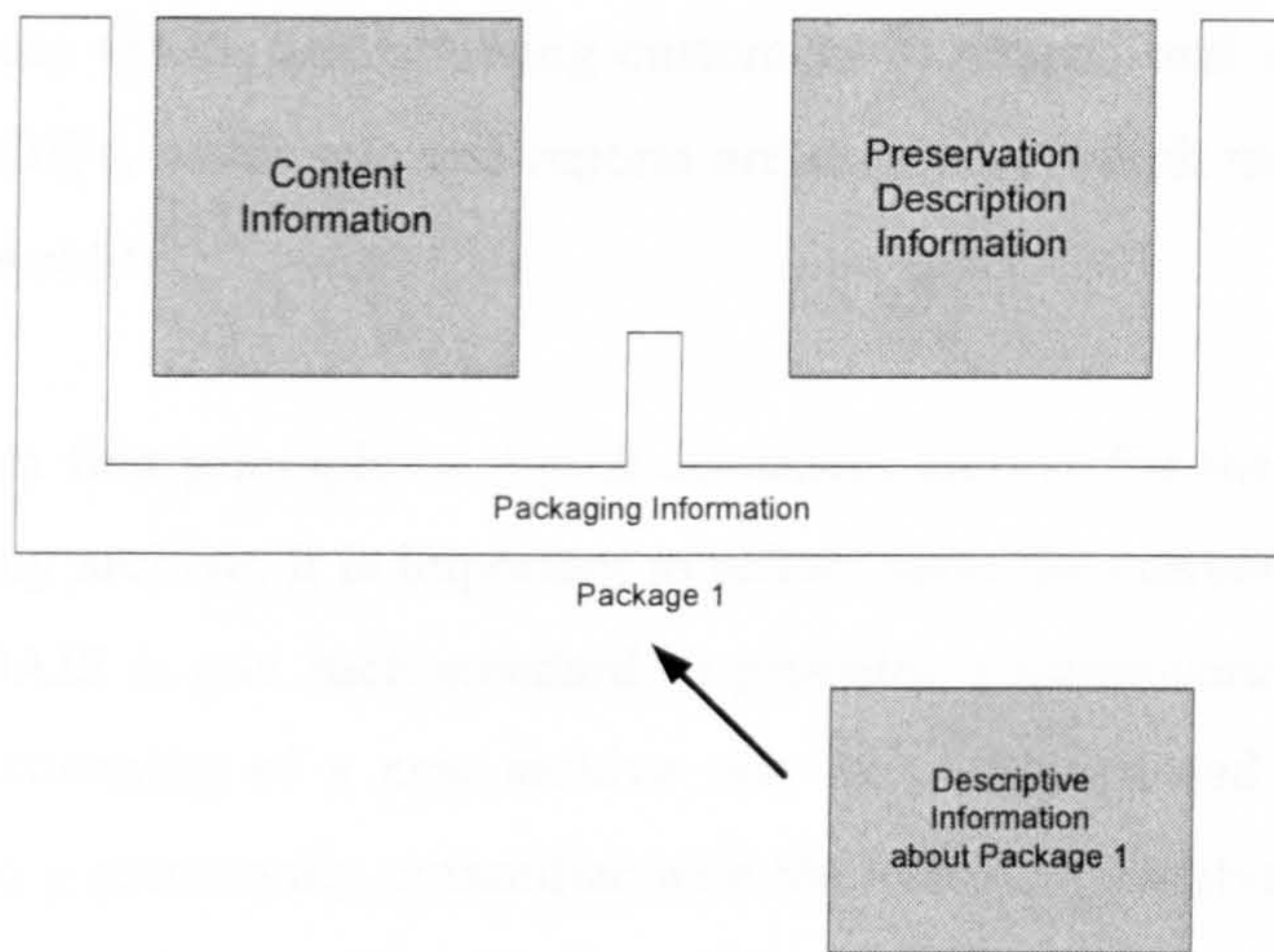


Figure 6.3: Information Package Concepts and Relationships

The function of the six functional entities is as follows.

- **Ingest:** provides the services and functions to accept SIPs from Producers and prepare the contents for storage and management within the archive. The CCSDS publish a specification for the negotiation which may take place to submit information between the Producer and the Archive in [CCSDS, 2004].
- **Archival Storage:** provides the services and the functions for the storage, maintenance and retrieval of AIPs.
- **Data Management:** provides the services and functions for populating, maintaining and accessing both Descriptive Information which identifies and documents archive holdings and administrative data used to manage the archive.
- **Administration:** provides the services and functions for the overall operation of the archive system. It also provides additional operations such as inventory management, report generation and migration/updating archive contents.
- **Preservation Planning:** monitors the environment of the OAIS with the essential objective of ensuring knowledge maintained within the archive remains accessible to the Designated User Community over the long term, even if the original computing environment becomes obsolete.
- **Access:** provides the services and functions that support Consumers in determining the existence, description, location and availability of information

stored in the OAIS, and allowing customers to request and receive information products. DIPs, result sets and reports are responses which may be generated by the Access entity.

To design from first principles a digital document archive for the Semantic Web or, more generally, any archive, it is important to reflect upon the current relevant standards in this domain. OAIS is one such standard. It provides a framework against which the structure and functionality of a new archive may be compared and contrasted. It also makes available to a community unfamiliar with the details of archives, in particular the Semantic Web community, considerable experience and knowledge so that in the design of a new archive implementation, a mature perspective of the issues involved may be considered.

6.4 Open Archive Initiative (OAI)

According to its mission statement, *the Open Archives Initiative develops and promotes interoperability standards that aim to facilitate the efficient dissemination of content. The Open Archives Initiative has its roots in an effort to enhance access to e-print archives as a means of increasing the availability of scholarly communication*¹.

Of particular interest is a framework for metadata harvesting produced by the OAI called the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [Lagoze & van de Sompel, 2001]. Its importance is to define a protocol, which has been widely adopted, to enable access to metadata by consumers to digital archives. OAI-PMH defines a **repository** which is a network-accessible server that can process the 6 OAI-PMH requests and a **harvester** which is a client application that issues the requests. Communication is effected by HTTP GET or POST methods and the response must be well-formatted XML. An example request by HTTP GET is:

```
http://an.oa.org/OAI-script?  
verb=GetRecord&identifier=oai:arXiv.org:hep-th/9901001&metadataPrefix=oai_dc
```

Here, the request type or verb is GetRecord. There are 6 such verbs which together define the capability of OAI-PMH

¹ See www.openarchives.org

- **GetRecord:** retrieves an individual metadata record from a repository. There are two arguments present besides verb, namely `identifier` which is a unique identifier, a Uniform Resource Identifier (URI) for a resource and `metadataPrefix` which identifies the metadata to be returned with the resource (see below).
- **Identify:** retrieves information about a repository such as a human readable name for the repository or its base URL.
- **ListIdentifiers:** an abbreviated form of `ListRecords` retrieving only headers rather than records. Header information includes the unique identifier for a record and a datestamp.
- **ListMetadataFormats:** retrieves the metadata formats available from a repository. These formats are described in XML Schema and for each format available, the URI of the schema is returned together with a metadata prefix, a key string which may be incorporated into other calls such as `GetRecord`.
- **ListRecords:** used to harvest records from a repository. Optional arguments permit selective harvesting of records based on set membership and/or datestamp. `metadataprefix` is also a required argument.
- **ListSets:** retrieves the set structure of a repository. A set structure, as a simple example, may attribute resources to be either *music* or *video*. Each set structure enables selective harvesting for `ListIdentifiers` and `ListRecords` verbs.

Indeed it is worth noting at this point the significance of sets within this protocol in relation to the broader picture of the work in this document. A significant subset of sets may be considered as classifications of resources within the archive and for the typical archive with which we are concerned, these resources are documents. So, in the context of a standard for interoperability of archives, sets or classifications are deemed to be important and in the context of document archives in general, effective methods of automatic classification will prove to be most valuable.

6.5 The Dublin Core Metadata Initiative (DCMI)

The Dublin Core Metadata Initiative is *an organization dedicated to promoting the widespread adoption of interoperable metadata standards and developing specialized metadata vocabularies for describing resources that enable more intelligent information discovery systems*. It maintains a Web site with comprehensive information about its mission at dublincore.org. It is an effort towards making it easier to find information on the Internet through the following activities.

- 1. Developing metadata standards for discovery across domains.
- 2. Defining frameworks for the interoperation of metadata sets.
- 3. Facilitating the development of community- or disciplinary-specific metadata sets that are consistent with items 1 and 2.

In particular, it has defined a set of 15 descriptive semantic definitions of metadata elements in the Dublin Core Metadata Element Set (DCMES). These elements are listed in Table 6.1 and are meant to enable the representation of key types of information found within information systems.

Title	Creator	Subject	Description	Publisher
Contributor	Date	Type	Format	Identifier
Source	Language	Relation	Coverage	Rights

Table 6.1: Simple Dublin Core Metadata Element Set, Version 1.1

The 15 elements¹ make up Simple Dublin Core. Qualified Dublin Core includes an additional element, Audience, as well as a group of element refinements (also called qualifiers) that refine the semantics of the elements in ways that may be useful in resource discovery.

¹ Full details of these elements may be found at dublincore.org/documents/dcmi-terms/

Dublin Core is mostly represented in HTML¹, RDF/XML and XML. For example, Figure 6.4 shows some Dublin Core metadata represented in RDF/XML syntax to give additional information about a resource, namely, an audio recording of a guide to growing rose bushes. This example illustrates the benefits introduced by a metadata initiative such as Dublin Core. Not only does it enable the publication of metadata in both a human readable and machine processable format on the Web, but it also stresses the importance that such metadata sets become universally accepted standards so that the complexity of having innumerable ad hoc and proprietary formats is avoided.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">

  <rdf:Description rdf:about="http://media.example.com/audio/guide.ra">

    <dc:creator>Rose Bush</dc:creator>
    <dc:title>A Guide to Growing Roses</dc:title>
    <dc:description>Describes process for planting and nurturing
different kinds of rose bushes.</dc:description>
    <dc:date>2001-01-20</dc:date>

  </rdf:Description>
</rdf:RDF>
```

Figure 6.4: Some metadata in Dublin Core formatted within RDF/XML

6.6 Summary

This chapter has outlined a number of key factors and standards which should be considered when designing a document archive for the Web and, in particular, the Semantic Web. The nature of the Web today is such that much of the information is transient and little care is taken to establish the consistency and viability over time of resources maintained there. The archive community reminds us that this inheritance needs to evolve if the Web is to succeed in becoming a long term reliable provider of knowledge. Factors such as integrity of information, risk management and knowledge management and preservation must be awarded first class citizenship in the design of Web based knowledge bases and digital archives of the future.

¹ See www.ietf.org/rfc/rfc2731.txt

OAIS provides us with a conceptual model against which actual archive systems may be designed and implemented and later compared and contrasted. Currently actual archives on the Web that are either informed by or actually conform to OAIS do just that: they integrate with the Web as opposed to the Semantic Web. Such a system is DSpace¹ [Smith et al., 2003], an OAIS compliant open source digital repository system from Massachusetts Institute of Technology (MIT) Libraries and Hewlett-Packard Labs. Simile² is a substantial effort to enhance DSpace to give it support for arbitrary schemata and metadata, primarily through the application of RDF and Semantic Web techniques. The challenges facing such an effort are likely to be quite steep, in reality, due to the lack of a mature Semantic Web and technical issues such as the underlying differences of the data models: conventional archive systems typically use a relational model whereas current practice for the Semantic Web implies a knowledge base making use of descriptive logics as its underlying representation model.

Interoperability of archives is necessary. A user wishes to use a single client browser to access metadata from different libraries, archives and museums all over the world. This is the objective of initiatives from the OAI. Indeed interoperability such as this can be considered one of the key benefits promised by the emergent Semantic Web. Further, OAI-PMH may in fact be considered a specification for the requirements of an initial implementation of archives for the Semantic Web: an ontology may be designed based on the key concepts presented by the protocol (records, metadata, sets, etc) and a set of Semantic Web services designed to fulfil the functionality of the 6 verbs described above.

Since a key component of the Semantic Web is its capability to represent metadata, it is unsurprising that initiatives which create metadata standards such as that achieved by Dublin Core are important. The Dublin Core elements are themselves published as a set of RDF properties³. Its importance in connection with this work is twofold. Firstly, Dublin Core elements describe fundamental properties of networked resources such as title, description and language. These properties are directly applicable to documents in archives such that reusing these properties in an ontology for archives is

¹ See dspace.org

² See simile.mit.edu

³ See purl.org/dc/elements/1.1/

likely to be constructive. Secondly, as a standard for metadata in itself, Dublin Core demonstrates the importance of universally accepted descriptions of metadata for the interpretation of machines as well as by human users.

Chapter 7 Ontologies

7.1 Introduction

In Chapter 5, ClassAKT was developed as a Semantic Web service which classifies Computer Science related documents according to the ACM classification hierarchy. It makes use of the AKT Reference Ontology¹ to ontologically ground its inputs and outputs. The AKT Reference Ontology was designed to model university research departments and it is a very interesting ontology for research purposes. However, it is felt to be very general in scope and designed to examine the properties particular to large standalone knowledge bases (mining or visualizing knowledge for example) as opposed to considering functionality that provides interaction with other artifacts on the Semantic Web, in particular Web services. It was therefore felt to be inappropriate for the further development of ontologies in which to ground further iterations of ClassAKT. Indeed it is argued that principles common to component-based development in mainstream software engineering remain valid in ontology use and development in the Semantic Web, that is:

- Develop generally applicable components of a reasonable size and reuse them across systems.
- Make sure components are adaptable to varying contexts.

To this end, this chapter introduces the design of ontologies which can be regarded as fundamental inasmuch as they define a generic domain of interest while being extensible to allow further refinement for a specific application as required. In particular, ontologies

¹ See www.aktors.org/ontology/portal

are developed that describe the domains of both a digital archive and a framework of classification techniques.

7.2 An Ontology for a Digital Archive

In the previous chapter, it was argued that although there are numerous varieties of information institution, they may be generalized into three types, namely museums, libraries and archives. We now consider the development of ontologies to describe these institutions. As core repositories of knowledge, they share a number of properties which will make their ontologies look similar, as outlined in the previous chapter. Such properties are common descriptions of certain metadata or techniques for knowledge management. However, there also exist a number of differences which will inevitably lead to a requirement that different ontologies will be required for different information institutions. For example, the metadata associated with some physical object in a museum may share some similarities with that required to describe an electronic document on the Internet but it will at a detailed level be different.

To this end, what is required is a complete and extensible ontology that expresses the basic concepts common across a variety of domains and media types and that can provide the basis for specialization into domain-specific concepts and vocabularies. Indeed, such an ontology has been developed in the form of the ABC ontology [Lagoze & Hunter, 2001, Hunter, 2003]. The ABC ontology has been developed within the Harmony international digital library project¹ which ran between 1999 and 2002 and was funded by DSTC (Australia), JISC (U.K.) and the NSF (U.S.). The role of such a *core ontology* is outlined in Figure 7.1. Here, the CIDOC CRM ontology [Doerr et al., 2003] and ON9.3 ontologies may be ‘retrofitted’ to integrate with the ABC ontology. The MPEG-7 and MPEG-21 ontologies were purpose built to exemplify use of the ABC ontology whereas the digital archive ontology was designed for use in this thesis.

The ABC ontology addresses the essential underlying concepts which are required to model any information institution. Hunter argues that such an ontology must be capable of answering queries with advanced semantics such as “who was responsible for what, when and where”. That is, a metadata model must provide a logical foundation for

¹ See <http://metadata.net/harmony/>

temporal semantics and consistent attachment points for the agents, actions and situations involved in these temporal semantics.

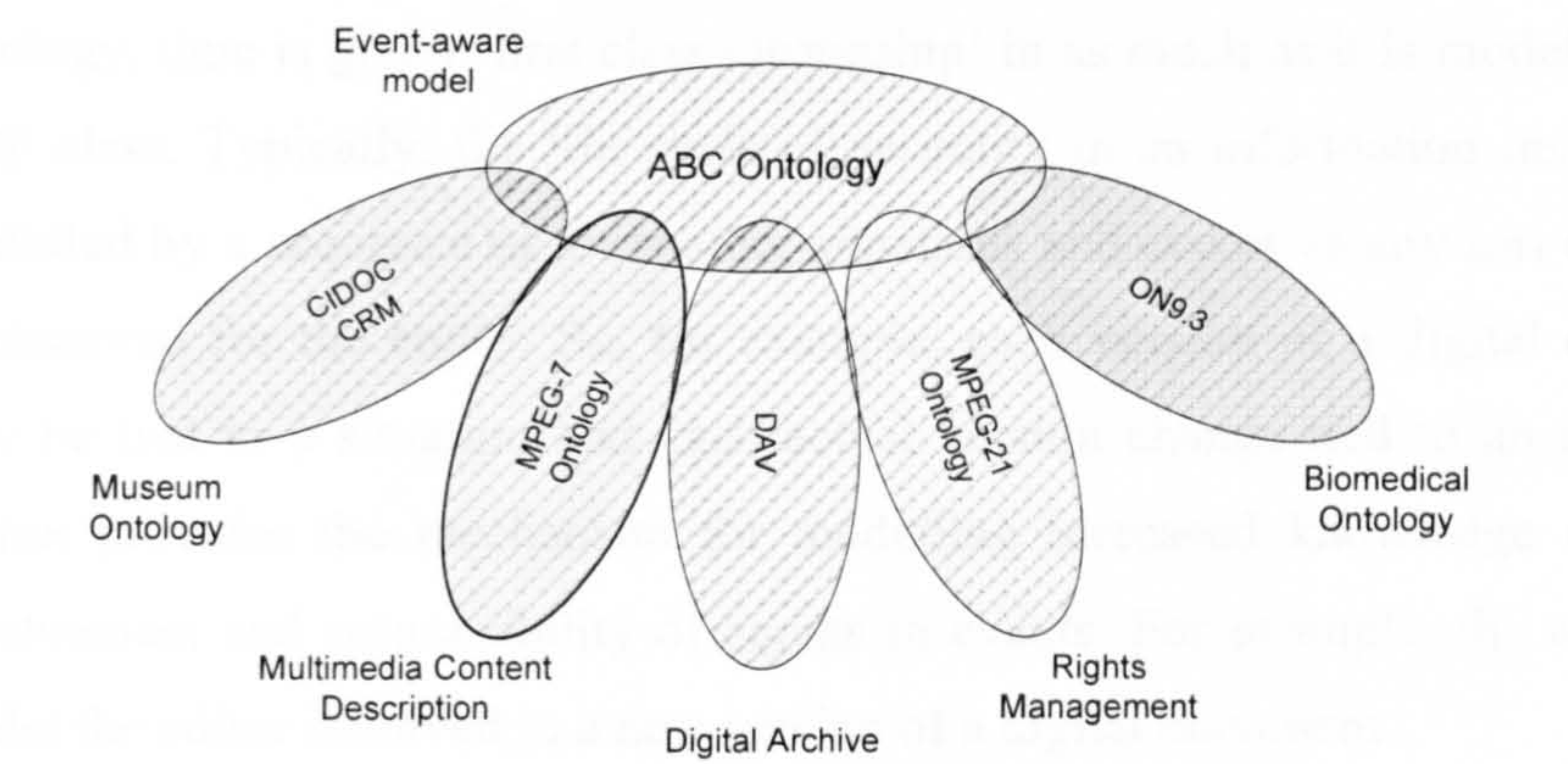


Figure 7.1 The ABC Ontology as a Core Ontology

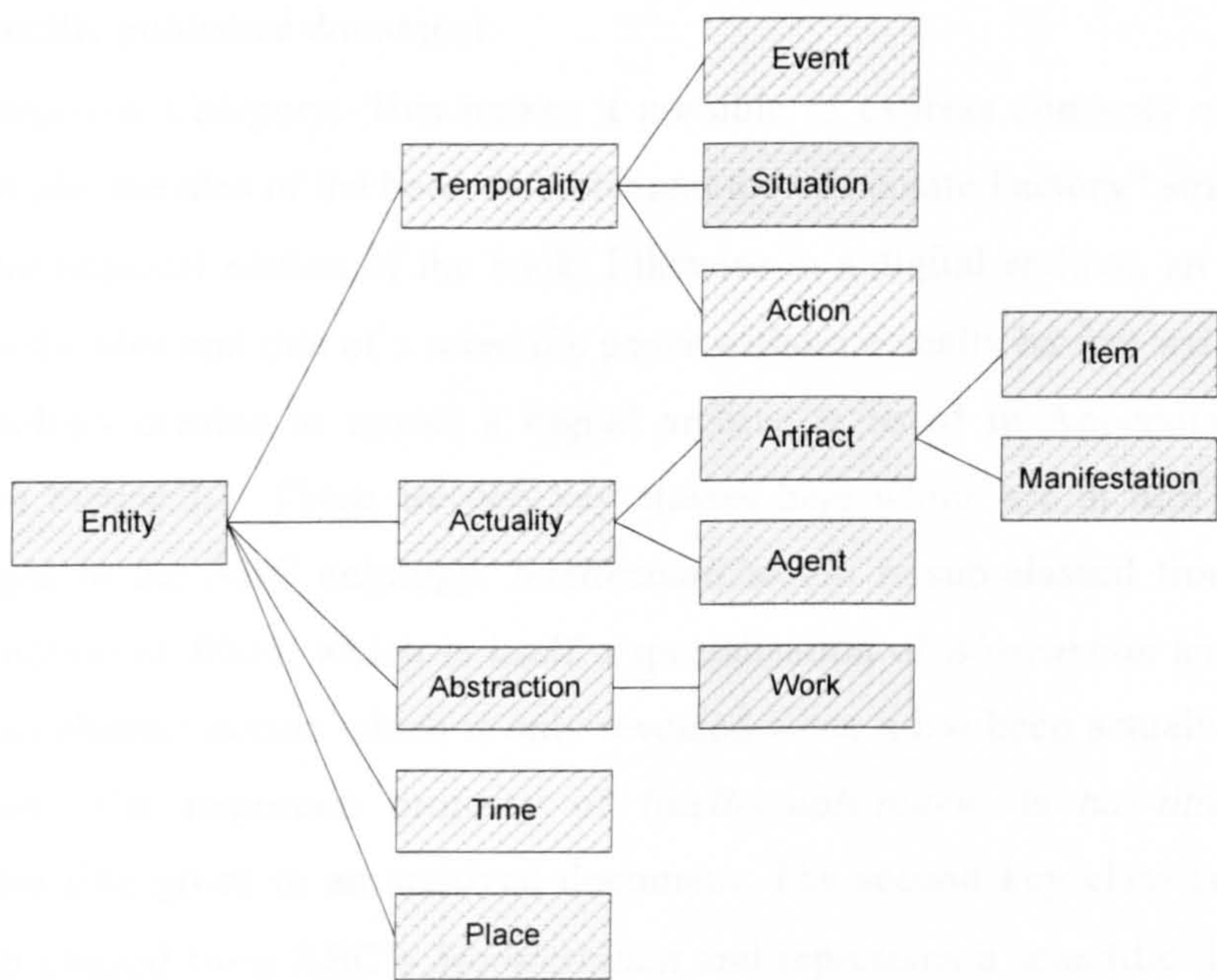


Figure 7.2 The ABC Class Hierarchy

The ABC ontology is available at <http://metadata.net/harmony/ABC/ABC.rdfs>. The class hierarchy is illustrated in Figure 7.2 and the function of each of the main classes is as follows:

- *Temporality Category*. This may be considered by analogy with the well known state transition diagram. A state corresponds to ABC's concept of a *Situation* whereas a transition could be considered equivalent to an *Event*. In the ABC ontology, time is given 'first class citizenship' in as much as it is modelled by an RDF class. Typically, the life cycle of an entity in an information institution is modelled by a sequence of alternating situations and events as any sort of change is observed for the entity. So, for example, each version of a digital document may be tied to a situation and each actual version change tied to an event. An *Action* provides the mechanism for modelling increased knowledge about the involvement and responsibility of agents in events. For example, the action can model the editor involved in a new version of a digital document.
- *Actuality Category*. This models a tangible entity, something which actually exists. In a museum this may be a vase or a dinosaur bone. In a digital archive it is a specific published document.
- *Abstraction Category*. This makes it possible to express concepts or ideas. For example, the idea of the book "Charlie and the Chocolate Factory" stripped of any actual physical edition of the book. Likewise in a digital archive, an author may have the idea and title of a scientific paper without actually having written it yet.

The ontology created to model a digital archive is listed in Appendix B¹ and is illustrated in Figure 7.3. There are two key classes here which are in turn sub-classed from concepts in the ABC ontology. *IntellectualContent* is sub-classed from the ABC ontology's notion of *Work*, which is itself a specialisation of *Abstraction* and addresses the idea of an abstract notion which is only revealed when it has been actualized in some *Manifestation*. The important property of *IntellectualContent* is *has-title* which is invariably the title given to an archived document. The second key class is *Document* which is sub-classed from ABC's *Manifestation* and represents a scientific document in an archive addressable via a URL. *Document* has two key datatype properties, namely *has-location* which is the URL address of the document and *has-format* which indicates the document's MIME type, a key requirement during classification where for example, machine processing a file requires the knowledge as to whether it is text, PDF or some

¹ At time of writing, see also http://classakt.aktors.org:8000/ontologies/digital_archive.owl

other format. Not shown in Figure 7.3 is that the datatype properties described above have in fact been sub-typed from appropriate properties in Dublin Core with the objective of giving these properties in effect stronger typing.

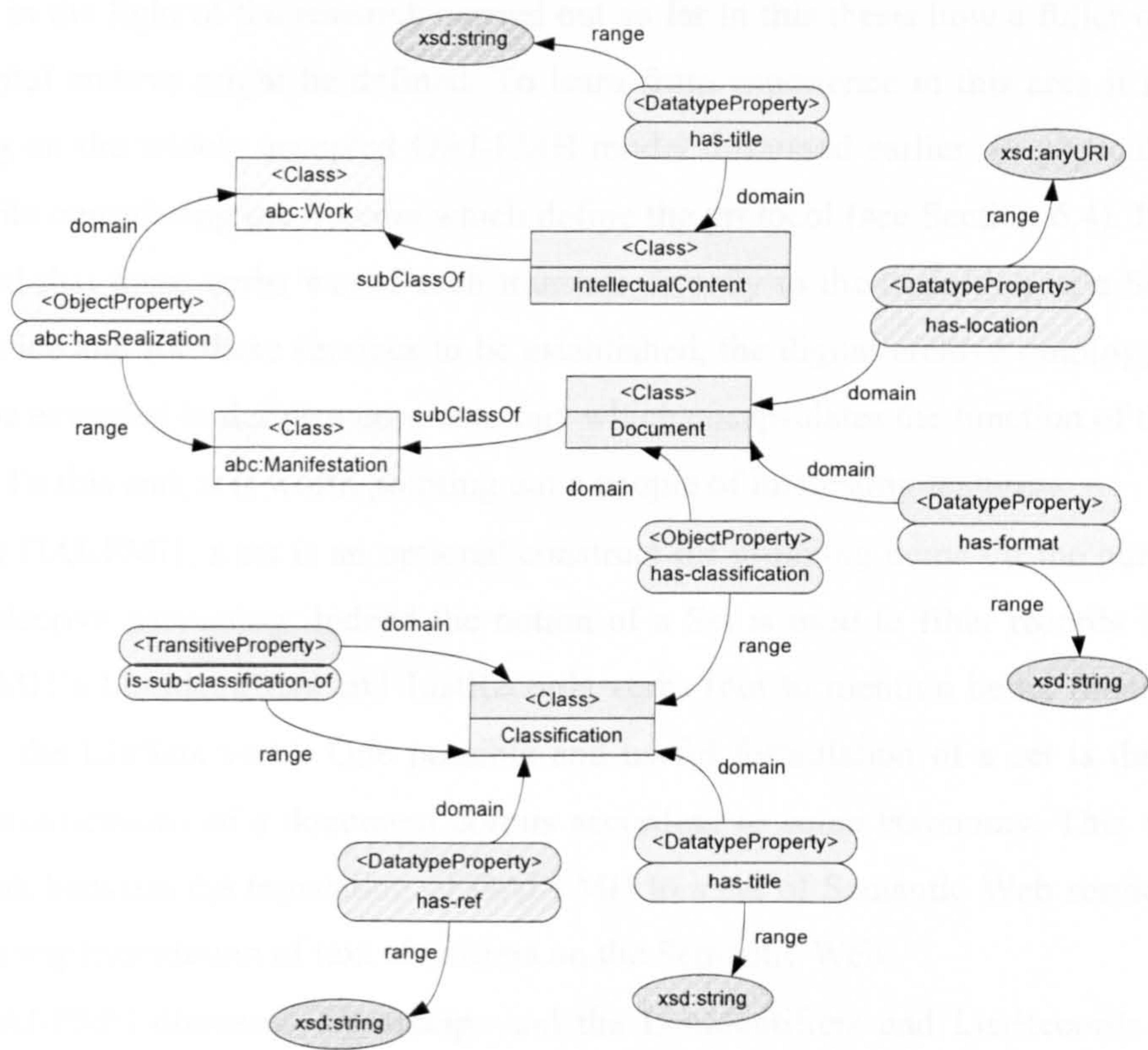


Figure 7.3 An Ontology for a Digital Archive

In information institutions in general, searching and browsing are fundamental, useful activities for the retrieval of pertinent materials. Certainly, an inventory classified according to some scheme is a key enabler of these two processes. In addition, important to the idea of a digital archive within the context of this work is (clearly) the notion of classification and in particular the notion of automated text classification. To these ends, *Document* has an object property called *has-classification* with a range of *Classification*. *Classification* is given two datatype properties *has-ref* and *has-title* which together define

a unique class. For example, from the ACM classification scheme, *has-ref* may be ‘A.’ and *has-title* would then be ‘General Literature’.

To date, the role of temporal semantics has not been fully investigated in the context of the digital archive ontology as it is not a key requirement to consider within the context of this work which focuses on text classification. However, it is interesting to consider in the light of the research carried out so far in this thesis how a fuller ontology for a digital archive might be defined. To learn from experience in this area it is worth reflecting on the widely accepted OAI-PMH model discussed earlier. In particular, it is worthwhile considering the 6 verbs which define the protocol (see Section 6.4). It can be speculated that these verbs would each translate directly to the definition of a Semantic Web Service and for these services to be established, the digital archive ontology would have to be extended to define a core ontology which encapsulates the function of the Web services. To this end, it is worth pointing out a couple of interesting features:

- In OAI-PMH, a set is an optional construct for grouping items for the purpose of selective harvesting. Indeed the notion of a Set is used to filter records in OAI-PMH’s ListIdentifiers and ListRecords verbs (not to mention being fundamental to the ListSets verb). One possible and useful formulation of a set is the set of classifications of a document corpus according to some taxonomy. This offers a link between the translation of OAI-PMH to a set of Semantic Web services and an implementation of text classifiers on the Semantic Web.
- OAI-PMH discusses timestamps and the ListIdentifiers and ListRecords enable selective harvesting of archive records between two points in time, the ‘from’ and ‘until’ timestamps encoded according to ISO 8601¹. It talks about timestamps of records which may be modification, creation and deletion and it would be fairly logical to sub-class each of these from the ABC ontology’s *Event* class to introduce basic temporality into the digital archive ontology introduced here.

7.3 Extending the Digital Archive Ontology

The digital archive ontology introduced in Section 7.2 may be described as a core ontology in as much as it addresses the essential and generic features associated with a

¹ For an introduction, see www.iso.org/iso/en/prods-services/popstds/datesandtime.html

digital archive. Extending this ontology is possible and may be required by new users for various reasons. In particular, in this work there is a requirement to describe the ACM classification hierarchy and enumerate its possible classifications in order that there is a common definition available to all communities to refer to. This has been achieved in an ontology which extends the digital archive ontology and is (partially) listed in Appendix C¹. Figure 7.4 illustrates the operation of this ontology in conjunction with two made up papers each of which, notionally, has a single classification.

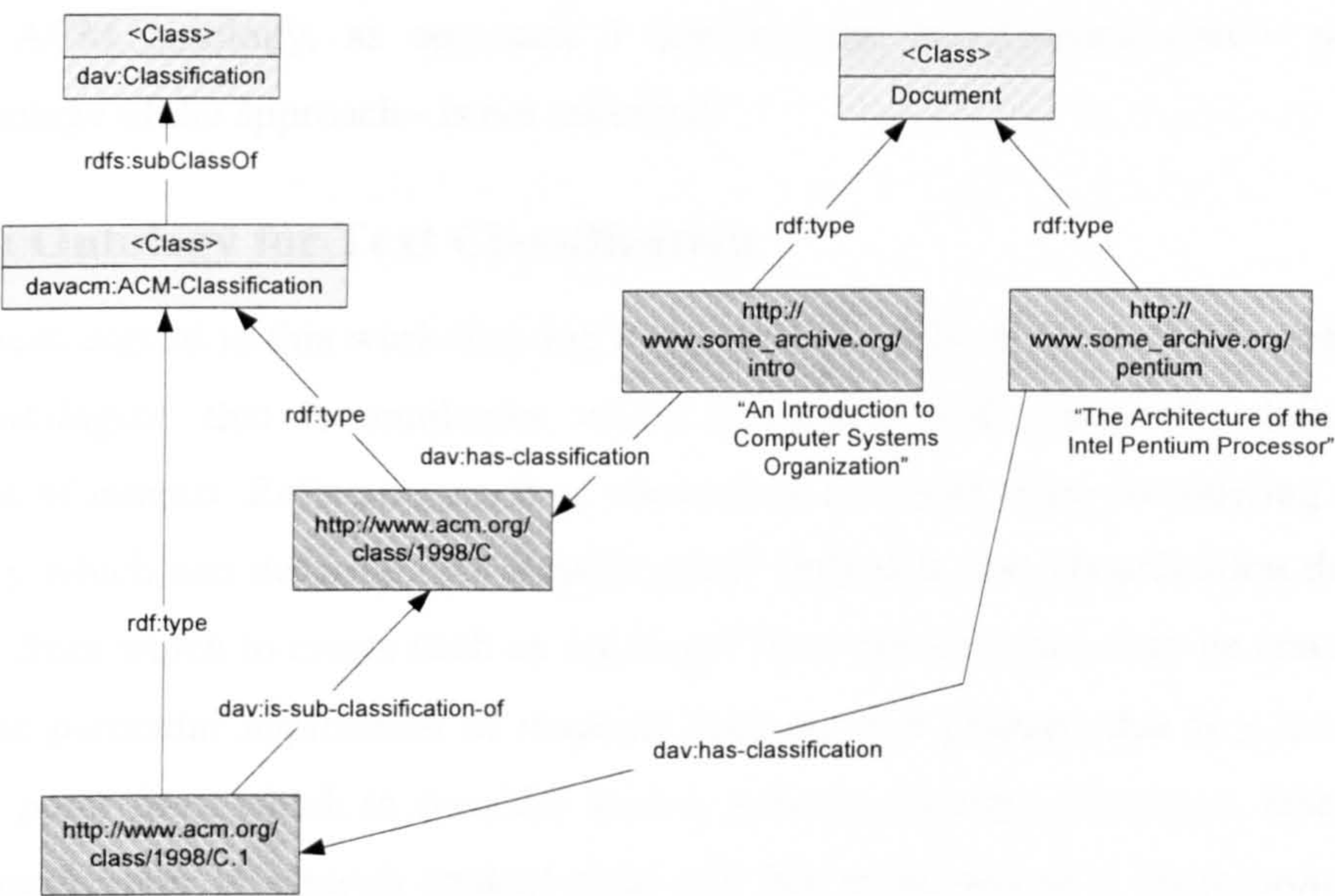


Figure 7.4 Mapping Documents to Classifications

Indeed the representation of classifications is not a trivial issue in itself. Intuitively, perhaps, enumerating the various classifications as instances makes much sense. However, the matter has received much wider attention within the W3C as can be observed in the document “Representing Classes As Property Values on the Semantic Web” [W3C, 2005]. The essential problem addressed by this document is that often it is logical to wish to use classes themselves directly as property values in OWL (or RDF) triples. For example, this would be the case if ACM classifications (for example C, C.1, etc) were sub-classed from the class ACM-Classification (instead of being instanced from

¹ At time of writing, see also http://classakt.actors.org:8000/ontologies/digital_archive_acm.owl

it as shown in Figure 7.4). The document goes on to show the various alternative approaches which may be taken to resolve this issue. Reflecting on this W3C document, the design of the ACM classification ontology introduced here may be considered to obey the pattern of approach 3 “Create a parallel hierarchy of instances as property values”. In light of this insight, the pros and cons of this design pattern may be clearly seen by studying the analysis of this approach outlined in the W3C document. In particular, the ACM classification ontology is compatible with RDF Schema and OWL DL (and hence OWL Lite). Additionally, by not strictly modelling a parallel hierarchy of class instances in the ACM ontology, as approach 3 implies, the maintenance cost – seen as a disadvantage of the approach - is not inherited.

7.4 An Ontology for Text Classification

It has been argued in this work that due importance must be given to the development of *core ontologies*, that is ontologies which accurately and generically define useful domains of interest. Reflecting on this, where does one start when considering a general ontology which can describe text classification? Indeed is text classification the correct domain from which to create such an ontology? Text classification may be considered as only one particular application of machine learning and perhaps that is a more useful starting point from which to consider such a generic ontology. However, research into text classification is a much studied field and has made use of a large proportion of available machine learning techniques. It provides a focused set of functionality to consider, in particular, two well known functional components, feature extractors and classifier generation, while allowing the key features of machine learning algorithms to be considered. Thus, in a Semantic Web poorly defined at the application level and utilising a complicated technology such as machine learning, perhaps it is a good place to commence an investigation.

In general we can identify two main users for services on the Semantic Web (or indeed the conventional Web of today) which affect text classification. Firstly, there are researchers. This is a difficult community to satisfy. In a service environment, it requires ‘plug and play’ capability of the various feature extraction and text classification algorithms which themselves are each likely to be parameterised in a unique, algorithm-

specific way. Researchers will of course be originators of modified and new algorithms which they will wish to be able to easily incorporate into a widely accepted framework. What is required is a tool built on top of an OWL-S services infrastructure which enables, most likely via a graphical interface, the composition of functional components which in turn leads to service composition and publication of the OWL-S ontologies. Secondly, there are clients who know nothing of text classification who wish to create and publish arbitrary classifiers of high quality on the Semantic Web having to hand only a tagged corpus. Indeed, satisfying these individuals' requirements may be considered the real objective of the work in this thesis. If automatically creating metadata is to be achieved on the Semantic Web, creators of this metadata will often have to be able to carry out this task with little or no knowledge of the underlying technology. To this end, a set of heuristics must be established to match a tagged corpus with specific characteristics (number of documents, document length, etc) with the most suitable classifier¹. If this can be achieved, it will make classification metadata significantly more accessible and consequently increase its usefulness and importance on the Semantic Web.

So what is required is an ontology for text classification which facilitates a service architecture that permits plug and play of its service interfaces (in effect, its inputs and outputs) so that feature extractors and classifiers may be interchanged in a service choreography as circumstances demand. In fact, a similar scenario has previously been addressed for Natural Language Processing (NLP) processing resources by a team at the University of Edinburgh [Klein & Potter, 2004]. The issues Klein and Potter encountered are quite similar to those that one finds in text classification. The NLP community has a wide variety of theoretical models for describing natural languages and it is difficult to achieve consensus in such an area of complicated technology. In addition there is an overriding requirement to decide on a general, useful framework which enables interoperability of NLP tools, the creation of an 'NLP Workbench'.

Figures 7.5 and 7.6 show the two key functional components, a feature extractor and a classifier generator, together with their principal inputs and outputs in a text classification framework. Whether the end user is a researcher or someone wishing to

¹ Indeed, it is worth pointing out that this is reflected in the empirical approach undertaken in Chapter 4. In Chapter 4, a framework of experiments was established which put forward an experimental approach for making a start in generating the heuristics discussed here in the context of users.

build a classifier, with little or no knowledge of the underlying technology, the process is essentially the same: a feature extractor must be matched with a classifier generator. In order to succeed in this matching, a number of constraints will exist between various inputs and outputs which must be taken into consideration if matching is to be successful and ‘plug and play’ of components is to be enabled. For example, the features generated by a feature generator must have undergone the same processing as the words in the tagged corpus of the classifier generator (for example, the words in neither or both must be stemmed). As a second example, the documents in the corpus input to either of the functional components may have to be of a specific format e.g. raw text may be acceptable but PDF may not. Indeed, in a posited workbench, a user can be imagined chaining together a number of additional services (PDF to text extractor, stemmers, etc) in order to make otherwise incompatible functional components work together.



Figure 7.5 Key Inputs and Outputs of a Feature Extractor

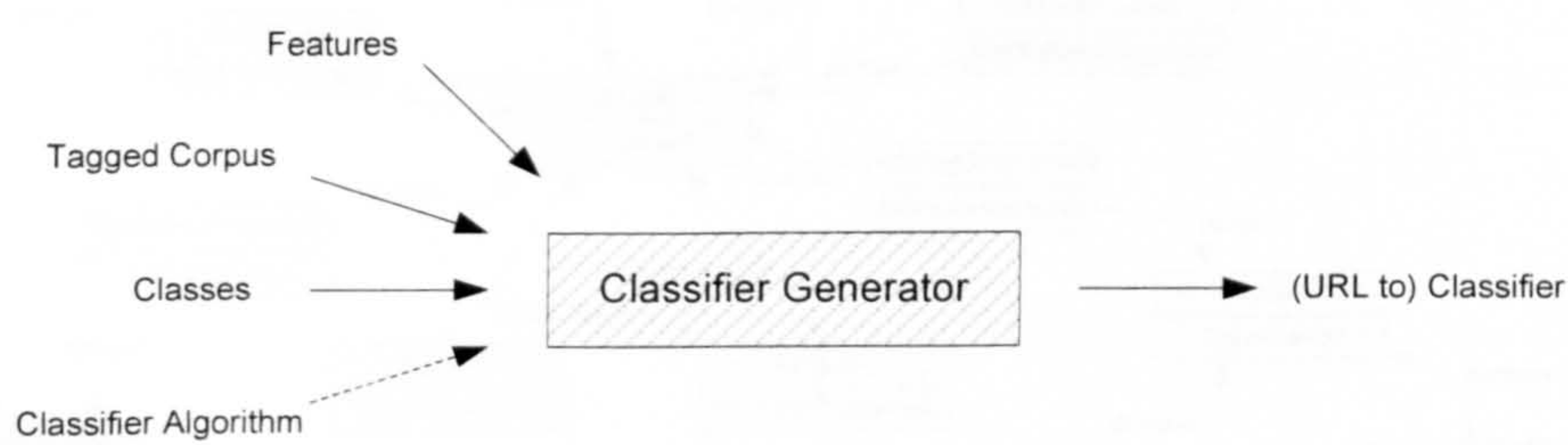


Figure 7.6 Key Inputs and Outputs of a Classifier Generator

From this discussion, a couple of key features emerge which should be taken into account when designing a suitable ontology for text classification. These two key features may be summarized as the design of the document and the design of the taxonomy of the functional components. Together, these features are incorporated into an ontology which

is listed in Appendix D¹. The following two sub sections describe the make up of this ontology in terms of these two key features in greater detail.

7.4.1 Design of a Document

Figure 7.7 is an ontological representation of a Document within the text classification ontology. A *Document* may be either a *WebDocument* or a *FeaturesDocument*. A *WebDocument* is a document which is employed to build, or be classified by, a classifier. Examples may be a Reuters news feed article or a newly published document that a user wishes to classify by the ACM hierarchy. A *FeaturesDocument* is a set of features produced by a feature generator or consumed by a classifier generator. It may be considered as a stream of word strings or features. Words in a *FeaturesDocument* and its associated *WebDocuments* will have constraints in place so that they share similar properties (for example, word stemming or language) so it is logical to place these document entities in the same framework in order that constraints may be established.

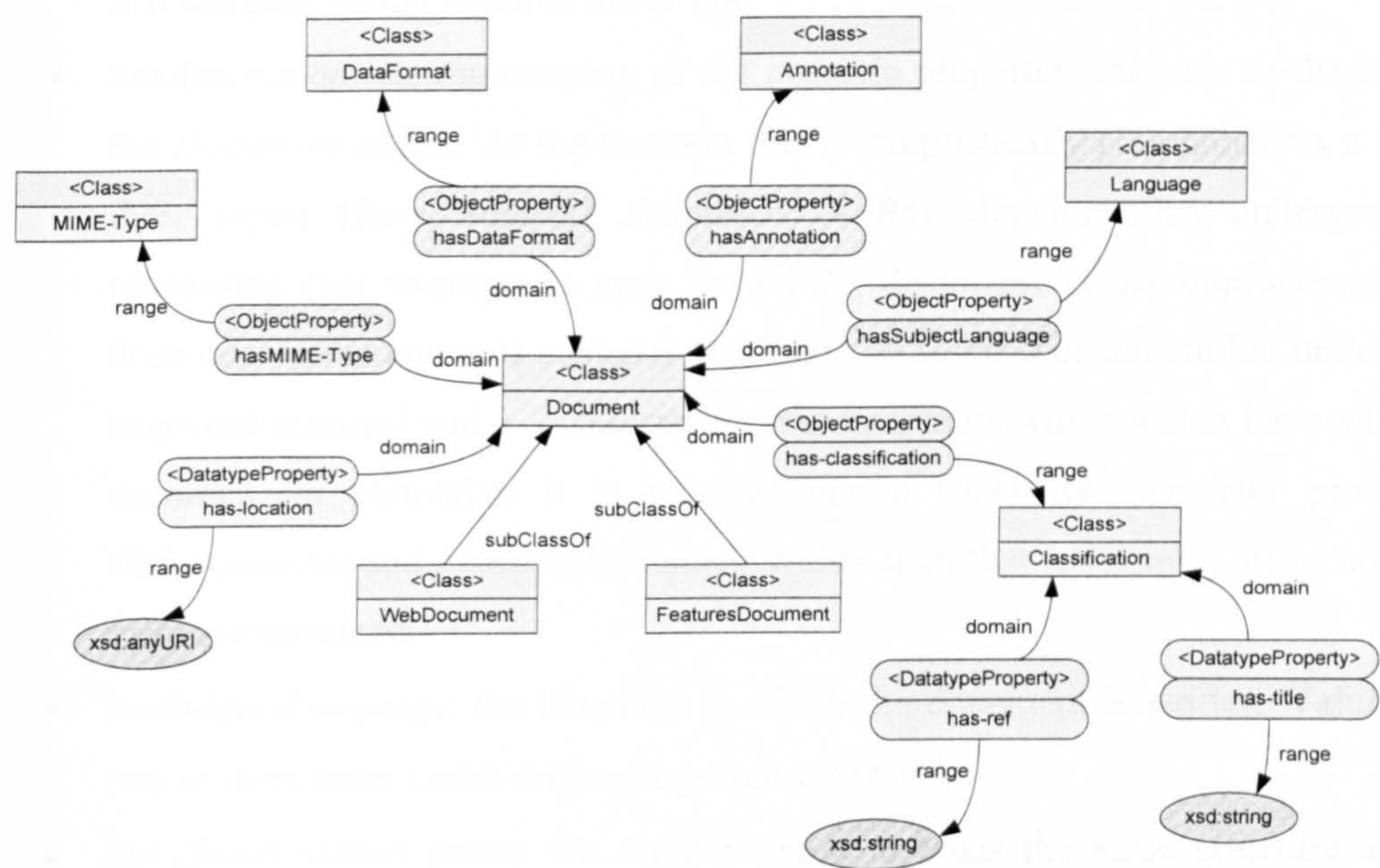


Figure 7.7 Document Representation in a Text Classification Ontology

¹ At time of writing, see also <http://classakt.aktors.org:8000/ontologies/TC-ProfileHierarchy.owl>

Details of the properties of the Document class are now explained in greater detail:

- *has-location*: a URI pointing to the location of the document on the Web.
- *hasMIME-Type*: the MIME type of the document in question. Of particular relevance in this work are the MIME types text/plain, text/xml and application/pdf. Feature Extractors and Classifier Generators will only be able to process documents of some small selective range of MIME types.
- *hasDataFormat*: a URI which points to a document describing the data format of the *Document* in question. It is typically an XML DTD or Schema. For instance, in the software architecture for text classification built for this work, a *FeaturesDocument* is an XML document which additionally describes the classifications the features are associated with. This is only one of many possible document formats. However by publishing the URI to the XML Schema of the features file as a property of the Document, a Classifier Generator may determine if it can process the features file or not.
- *hasAnnotation*: an enumeration of the possible properties intrinsic to the text of the *Document* object. At the moment this is simplistically maintained as a set of three types: {*Raw*, *Stopped*, *Stemmed*}. A *Raw* document has undergone no processing (for example it may be a PDF document or an unprocessed text document maintaining its sentence structure). A *Stopped* document has undergone stopword removal and a *Stemmed* document contains words which have all been stemmed. In particular, it is required that a Classifier Generator processes *WebDocument* and *FeaturesDocument* words such that uniformly, all or none of them are stemmed.
- *hasSubjectLanguage*: the language in which the document is written. Values are two or three letter codes originating from ISO 639¹.
- *has-classification*: serves the same purpose and has the same structure as the property of the same name in the ontology for the document ontology (see section 7.2). That is, a document would be expected to have one or more classifications

¹ See www.loc.gov/standards/iso639-2/

and each such classification has a ‘ref’ (e.g. in the ACM classification this might be ‘A.’) and a title (e.g. ‘General Literature’). Both *WebDocuments* and *FeaturesDocuments* may have classifications associated with them.

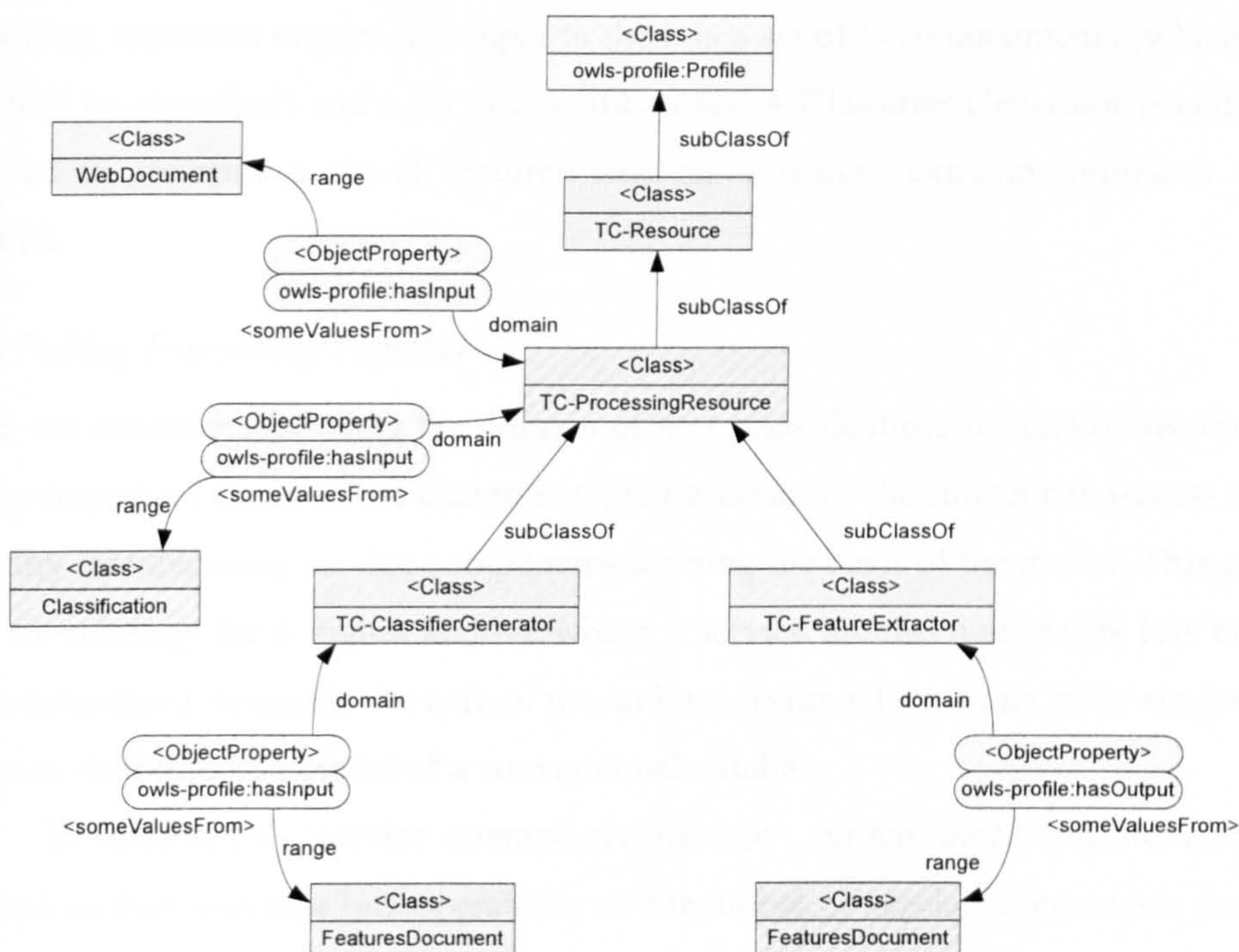


Figure 7.8 Processing Resource Representation in Text Classification Ontology

7.4.2 Design of a Taxonomy for Processing Resources

As discussed previously, Web services to effect text classification are implemented in this work utilising the OWL-S framework. Recall that the purpose of an OWL-S service profile is to advertise the service and to provide a description of its functionality in part through a description of the service’s inputs, outputs, preconditions and effects (IOPEs). In general it is possible to extend the core OWL-S ontology within a particular domain, in effect typing certain IOPEs in the service profile and attributing to them certain constraints. The result is that services may be advertised and requested in terms of their categorization. This is the approach described by Klein and Potter for NLP processing resources and adapted here in order to describe processing resources in text classification.

Figure 7.8 illustrates the core sub-classes in the text classification profile hierarchy as well as showing some of the constraints placed on key inputs and outputs pertaining to documents.

Figure 7.8 may be considered in conjunction with Figures 7.5 and 7.6. All processing resources require that input to them is a set of Web documents (which in each case will be classified) and a set of classifications. A Classifier Generator is additionally required to consume a set of features whereas a feature extractor produces a set of features.

7.4.3 Pulling Everything Together

When we consider modelling the domain of text classification, it quickly becomes clear that an important focus of the design is taking account of the functional aspects of what, in reality, are the many service components forming the basis of the model. This contrasts with an ontology for a digital archive where a service architecture merits less emphasis when considered alongside the core of the archive design which bears more similarities in design to the relational model of a conventional database.

To achieve this ‘service oriented architecture’, certain added considerations were required so that seamless interoperability of functional or service components was made possible. In particular, sub classing a ‘processing resource’ taxonomy from the OWL-S profile class enabled service parameters to be typed so as to permit plug and play of service components. Careful design of the properties of the key *Document* class further refined these service parameters to ensure IOPEs to services could be more tightly type matched.

Figure 7.9 demonstrates how service composition may take place. It is a fairly simplistic model focusing on the types of *Document* flowing through the system. ‘Information Gain Feature Extractor’ and ‘Naïve Bayes Classifier Generator’ are instances of *TC-FeatureExtractor* and *TC-ClassifierGenerator* respectively. ‘PDF to Text Extractor’ and ‘Word Stemmer’ are two additional functional components which are notionally part of, but not shown in, the architecture of Figure 7.8. The constraints imposed on the *Document* inputs and outputs play a key role in determining how the functional components may be connected together. Indeed, it is not difficult to imagine

the flow shown in Figure 7.9 being transposed, in essence, into the graphical user interface of a ‘Text Classifier Workbench’ tool in which functional components may be plugged together to produce useful composite processes. Such a tool would make use of a registry of Semantic Web service profiles to present the users with a tree of available service components.

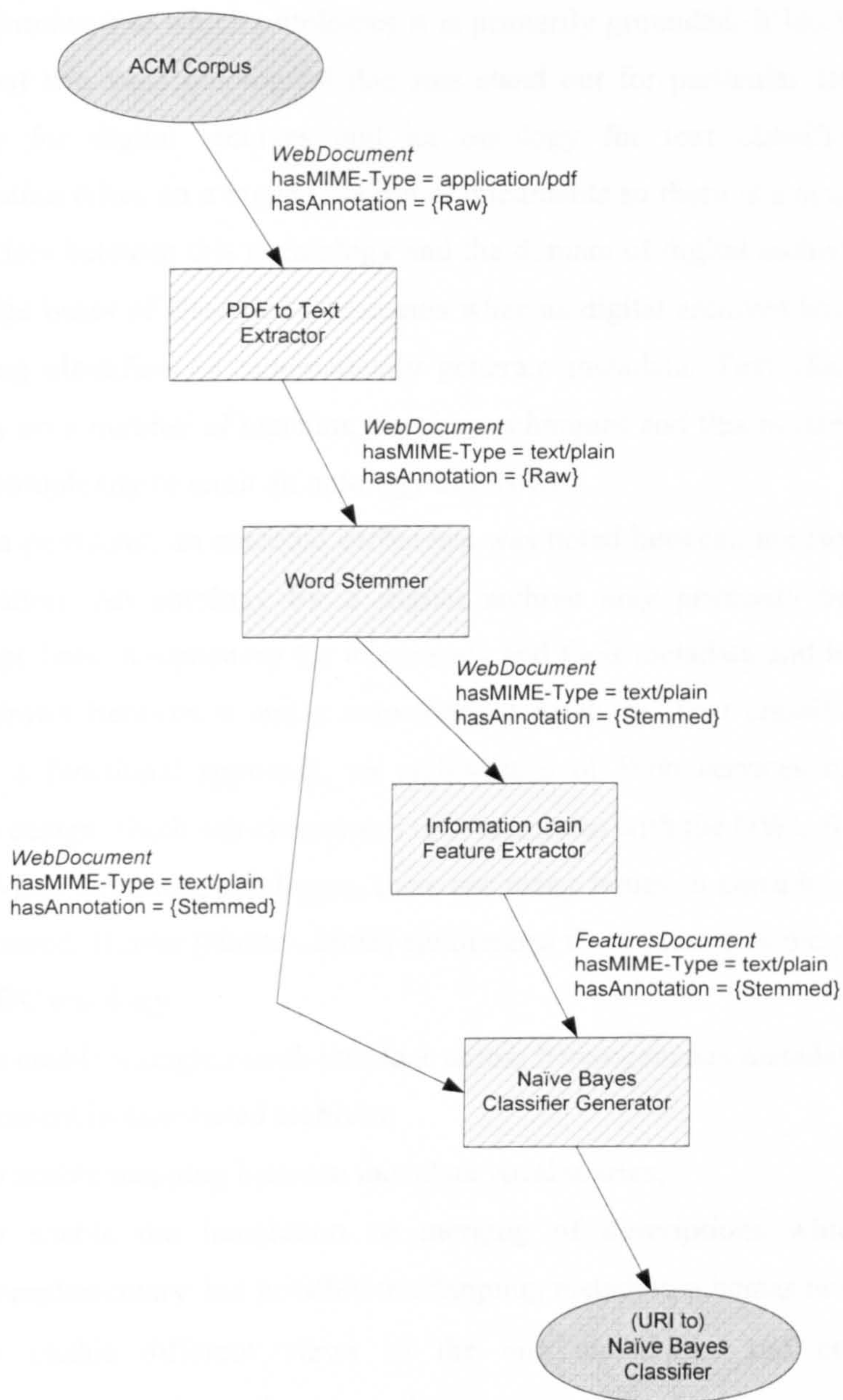


Figure 7.9 Document Types during Classifier Generation

7.5 Summary

Perhaps *the* key requirement to tie text classification into the world of the Semantic Web is to determine into which ontologies it is primarily grounded. It has been argued in this thesis that two such ontological domains stand out for particular attention, namely an ontology for digital archives and an ontology for text classification itself. Text classification relies on a tagged corpus of documents so there is a symbiotic relationship which arises between this technology and the domain of digital archives: classifiers need knowledge bases of classified documents whereas digital archives have much to gain by employing classifiers to automatically generate metadata. Text classification depends primarily on a number of machine learning techniques and this in itself suggests there is enough complexity to merit an ontology of its own.

In particular, an essential difference was noted between the two ontologies under consideration. An ontology for a digital archive may primarily be considered as a knowledge base, a repository for documents and their metadata and by nature analogies can be drawn between it and a conventional database. Text classification by contrast leads to a functional approach, an architecture of Web services underpinned by an ontology design which sub-classes and thus integrates with the OWL-S service profile.

When designing ontologies, there are many issues to consider if their value is to be maximized. Hunter [Hunter, 2003] enumerates these factors as they relate to her work on the ABC ontology:

- to enable a single search interface across heterogeneous metadata descriptions and content in distributed archives;
- to enable mapping between metadata vocabularies;
- to enable the integration or merging of descriptions which are based on complementary, but possibly overlapping, metadata schemas or standards;
- to enable different views of the one underlying and complete metadata description, depending on the users' particular interest, perspective or requirements.

In practice, a graphical tool may be postulated which enables visualization of various knowledge bases made available by the different information institutions, each sharing at

the core of their domain-specific ontology, the ABC core ontology. In respect of the ontology for text classification, a design pattern emerges in which the OWL-S service profile is sub-classed to effect plug and play of functional components. Generalization of this pattern may indeed be possible to enable the development of a general tool for visualization and composition of different ontologies.

Chapter 8 Web Service Generation

8.1 Introduction

Chapter 5 introduced a Semantic Web service, ClassAKT. This is a useful service to have available on the Semantic Web because generating classification metadata pertaining to the ACM taxonomy of Computer Science related documents is helpful for use in many applications. In particular, ClassAKT was developed with the object of generating additional metadata for a Triplestore associated with the AKT reference ontology. However, on a mature Semantic Web many useful taxonomies will be available and taxonomy specific services such as ClassAKT would ideally need to be created easily and ‘on the fly’ with no specialist technical knowledge required of the human implementer of the necessary services. This Chapter introduces such a ‘Web service generator’ for use in context with building text classifier services. It discusses the advantages and technicalities associated with such a framework and in particular outlines a solution to one particularly thorny issue, namely the requirement of making such a Web service generator work with a large number of independently designed ontologies in which the various taxonomies of documents are defined.

8.2 Web Service Generation

Today, applications of text classifiers on the Web occupy a number of related roles:

- *Web Classification*: filing Web sites or organizing search results under hierarchical directories. For example, Yahoo, DMOZ (dmoz.org).

- *Recommending (or Filtering)*: selecting documents which may or may not be relevant to a user, based on criteria of interest or appropriateness.
- *Detecting Unsuitable Content*: filtering out pornography or spam

As the Semantic Web takes shape, the requirement for effective text classifiers can only increase. Many tagged document corpora such as the ACM corpus exist for which an automatic classifier would be most useful. Indeed, publishing these corpora in a knowledge resource format (such as RDF or OWL) would not only represent a significant step forward for the Semantic Web but would make the case for building associated automatic classifiers compelling. The ability to generate metadata automatically and therefore cheaply (in terms of time and resources) for corpora using established taxonomies would facilitate simple yet essential features of search, filtering, grouping and browsing for arbitrary document sets.

One consequence of introducing Semantic Web technology is that it seamlessly facilitates the introduction of structure into the Web. Associating metadata with Web pages immediately offers the opportunity, for example, to build alternative search facilities for pages on the Web according to some criteria defined by metadata. This in itself may well provide a successful alternative to an established search engine such as Google with its reliance on text, in particular keywords, and relative ignorance of explicit semantics. In conjunction with this, Semantic Web technology makes it relatively straightforward to set up a knowledge base to represent some community (such as the AKT reference ontology¹ and associated Triplestore² for UK Computer Science departments). In this context, classifiers may be useful firstly to find pages on the Web which may usefully be referenced from the knowledge base and secondly, to associate classification metadata with sites, Web pages and documents facilitating search and related activities from within the knowledge base.

In short, the Web can be considered as a large collection of unstructured documents. The Semantic-Web-plus-automatic-text classifiers offer for the first time the opportunity of grouping document sets according to some criteria based on taxonomies in

¹ See www.aktors.org/ontology/portal

² See triplestore.aktors.org/

a uniform framework to facilitate semantic-based organization of the Web. For this vision to become a reality, work is required on two fronts:

1. Investigate, establish and publish comprehensive heuristics for corpora that lead to the implementation of effective text classifiers. (e.g. average document length, corpus size, key feature frequency, etc)
2. Investigate and introduce a functioning framework on top of Semantic Web technology to facilitate the widespread adoption of classifier technology such that useful applications may be created which make use of classification metadata.

Chapters 3 and 4 have been an initial effort towards (a) which at the least provides a baseline idea of effectiveness for classifiers built from corpora (Reuters-21578 and ACM) with very different textual properties. Published research on classifiers tends to concentrate on comparing and contrasting the performance of new types of classifier or variations of an existing classifier with established approaches. By producing the graphs of Chapter 4 where classifier performance is considered with variation in document or feature numbers, an initial effort is made to answer questions such as ‘if I have a corpus similar to Reuters, what is the minimum corpus size/number of features I need to assemble in order to implement a classifier of a given effectiveness?’. Having said this, publishing such heuristics in the public domain is something which may or may not happen. Today, if you have no knowledge in classifier technology but require it, you are likely to approach one of a number of commercial organizations that have developed their own software [Adams, 2001]. Without a concerted effort from within the research community, this is where such technical knowledge is likely to remain, as closely guarded commercial secrets.

Chapter 7 introduced ontologies which enable text classifiers to be tied into the Semantic Web thereby providing the groundwork for establishing (b). As mentioned previously, ClassAKT though a useful Semantic Web service for classifying ACM documents, is itself limited to classifying documents according only to the ACM taxonomy. In a mature Semantic Web, there will be numerous such taxonomies originating in their own independently defined ontologies. Just as the ACM digital library manages Computer-Science-related scientific PDF documents, other archives also exist for the vast majority of academic disciplines. As mentioned previously, it is useful to

classify Web sites and pages according to existing or new taxonomies. Indeed it may be useful to build classifiers from taxonomies constructed by combining partial document sets from various corpora. For example, a classifier may be built which sorts documents for different transport industries using existing corpora and associated taxonomies from various industry sectors such as automotive, aviation, shipping and so on.

Therefore, in the Semantic Web, functionality is required which can take some arbitrary taxonomy and corpus and generate a classifier which classifies documents according to this taxonomy. What is required is a ‘Web service generator’, a Semantic Web service that generates another Semantic Web service. Figure 8.1 illustrates the general situation.

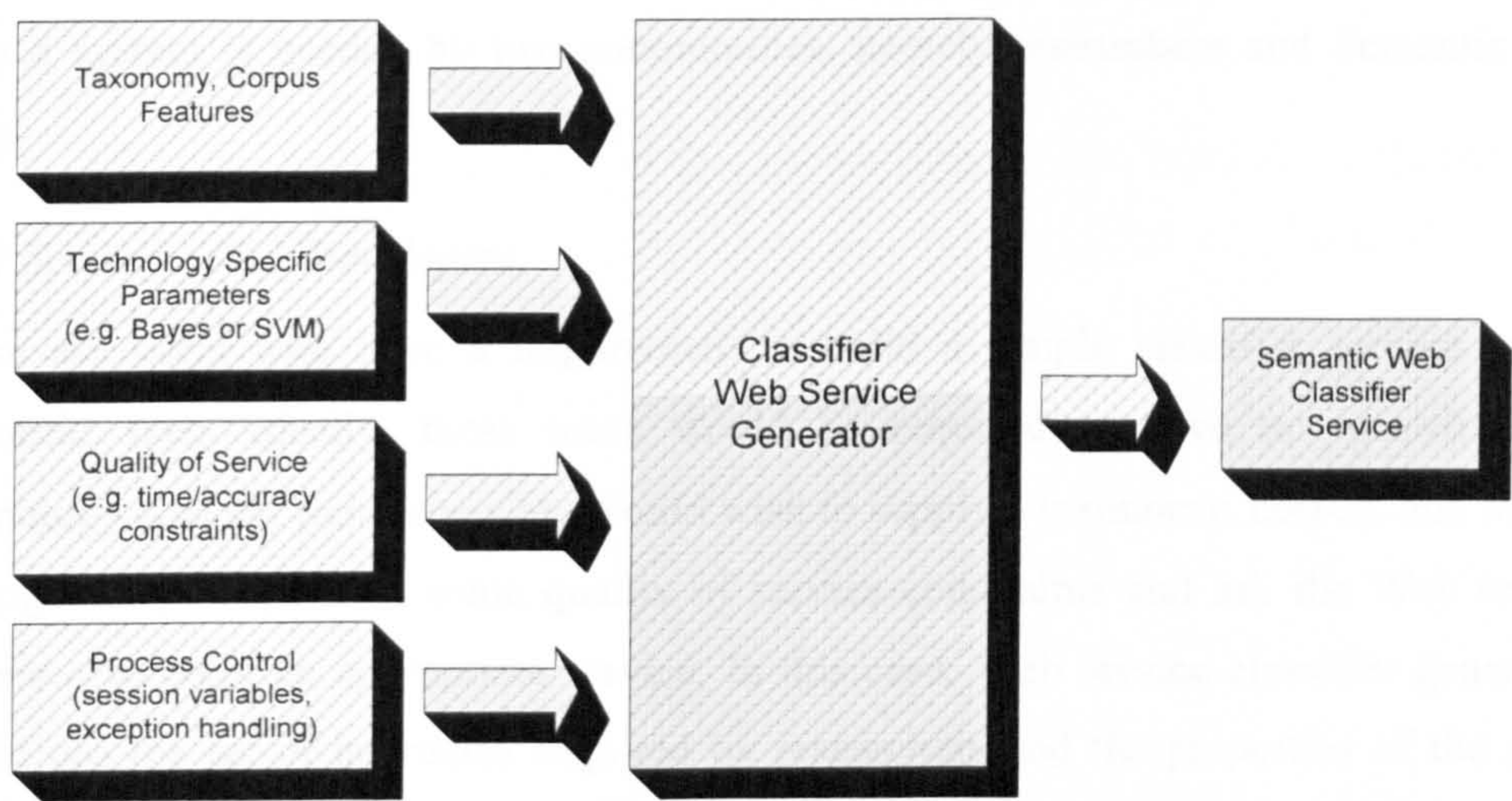


Figure 8.1 Semantic Web Classifier Service Generator

The Web service generator has a number of sets of inputs which are semantically independent of each other.

- Taxonomy, corpus and features which together form the source data set from which to construct a classifier.
- Classifier related parameters. These are not just which type of classifier to use (Bayes, SVM, etc) but, for each type of technology, which technology-specific parameters need to be selected (for example, which kernel to use for a SVM).

- Quality of service parameters. A user may wish to construct a classifier within a given time frame or have minimum requirements for classifier effectiveness (perhaps quoted in terms of a precision/recall breakeven point). Also, the user of the service may be charged for its use and semantics relating to pricing may have to be implemented.
- Process control parameters. In OWL-S terms, a Web service generator is a composite process. As such it requires coordination between atomic process invocations by means of a session variable. Other infrastructure is required for process control. In particular, a standardised framework for exception handling. To date, no known effort has been made towards standardising these issues within the OWL-S framework.

Such a system is needed by two communities, namely researchers and Semantic Web users.

8.2.1 Semantic Web Developers

These are users who have a requirement to build a simple classifier service on the Semantic Web. Ideally, these users would be required to have no knowledge of technology-specific parameters but would wish to supply a taxonomy, corpus, and feature set together perhaps with some quality of service constraints and ask the Web service generator to build an appropriate service. In this case, Web service classifier generators would use the set of heuristics acquired by researchers and the properties of the given corpus and would then build the best specification of a classifier for the given situation. The end product of their work would be text classifier Semantic Web services. These would then be published in registries or advertised/integrated in Web sites, to enable invocation by the general Web user in order to classify documents

8.2.2 Researchers

Researchers, in this case in Machine Learning, may wish to investigate classifier algorithms or make an empirical analysis perhaps in an analogous way to that undertaken in Chapter 4. For a researcher, technology-specific parameters are very important whereas specifying quality of service parameters is less so.

For research purposes it is perhaps more natural to make use of a standalone application like the one introduced in Chapter 4, in which the corpus exists on the same machine as the classifier software. In this case a researcher does not have to deal with service protocols or network latency caused by uploading a corpus onto a classifier machine. Certainly, the caching of corpora, on servers hosting classifier Web service generators, is likely to be an important optimisation. For example, the ACM or CiteSeer may wish to be hosts to classifier generators. However, one interesting possibility for researchers which might be able to take advantage of the ‘service orientated architecture’ introduced here is the Semantic Grid. This is discussed further in Section 8.6.

8.3 A Web Service Generator Implementation

8.3.1 Overview

Section 4.2 introduced a software framework comprising two standalone applications: *Features* which constructs a set of features given a tagged corpus and *Classifiers* which builds classifiers given a set of features and a tagged corpus. The source code of these two applications forms the code base of two new applications which are Web services, namely *FeatureCreationService* and *ClassifiersServiceEx*. These two new applications have been developed within the Microsoft .NET Web services framework and each one consists of a set of *WebMethods*. *WebMethods* are essentially discrete methods which may be called by Web clients of the service. Each of the two new services requires a set of these *WebMethods* which together define the method calls within a process, or in OWL-S terms, a composite process, which details the sequence in which the *WebMethods* are to be invoked. Sections 8.3.2 and 8.3.3 describe the processes for *FeatureCreationService* and *ClassifiersServiceEx* in greater detail.

Web services developed within the Microsoft .NET framework, when called with a particular URL, will return a WSDL interface for the service. Such WSDL interfaces may be consumed by the grounding of an OWL-S service description when the service is invoked. For each of the *WebMethods* defined in *FeatureCreationService* and *ClassifiersServiceEx*, an OWL-S atomic service interface has been defined. When ClassAKT was developed, tools to assist in the job of authoring and invoking OWL-S wrappers were unavailable. Consequently the task of creating such descriptions was error

prone and their use beyond demonstration purposes limited. Fortunately the situation has changed. OWL-S API¹ is a Java API for programmatic access to read, execute and write operations for OWL-S service descriptions. It is developed and maintained by Evren Sirin at the Mindswap group at the University of Maryland. At the time of writing, the latest release of OWL-S API was OWL-S 1.1.0-beta and it is this version which has been utilised to develop working, atomic, OWL-S Web service descriptions for the two Web service applications introduced here.

8.3.2 *FeatureCreationService*

Figure 8.2 is a composite service description showing the process in which four atomic OWL-S services are invoked to effect feature creation. The actual implementation of the OWL-S services occurs as four *WebMethod* implementations within *FeatureCreationService*². ‘Sequence’ and ‘Repeat-While’ are OWL-S constructs for composite processes (although an actual OWL-S composite service description has not to date been written).

`process_management.owl` is a stub ontology which in particular defines a session variable, `pm:session-id` the function of which is to allow the server to identify the composite process it is managing at a given time across atomic service invocations. The remainder of the inputs and outputs are defined in terms of the ontology for text classification, TC-ProfileHierarchy, introduced in Chapter 7. In brief, after a session is created, class descriptions followed by instance information (URL to a document plus its classification details) are sent to the server and then the service is again invoked to request a (URL to a) set of features, formatted as a *FeaturesDocument*.

¹ See www.mindswap.org/2004/owl-s/api/

² At time of writing, all OWL-S service descriptions (for example, the service *FeatureCreation_CreateSession.owl*) are published at <http://classakt.actors.org:8000/services/owl-s/1.1/>

xmlns:pm = http://classakt.aktors.org:8000/ontologies/process_management.owl#
xmlns:tc = http://classakt.aktors.org:8000/ontologies/TC-ProfileHierarchy.owl#

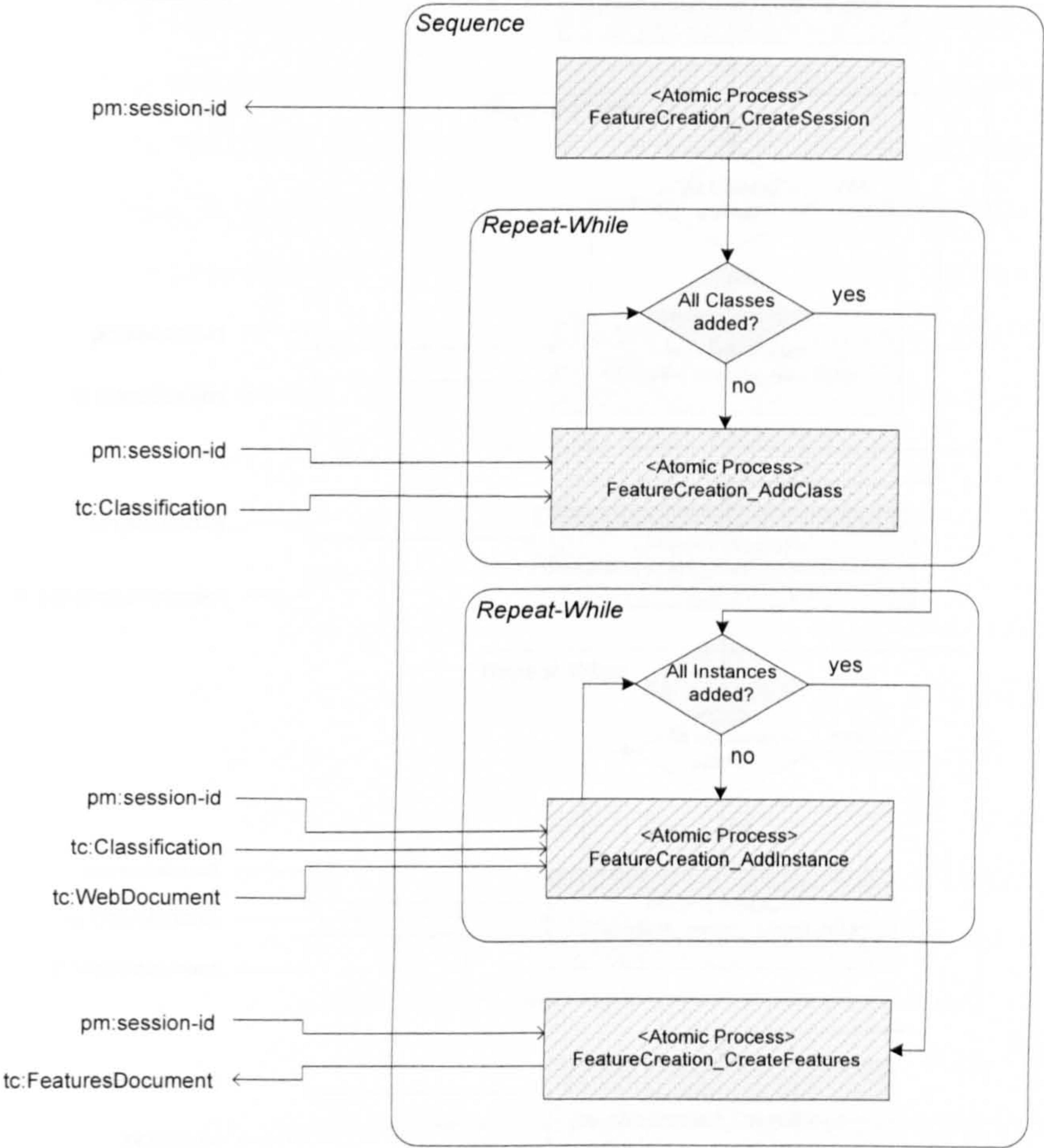


Figure 8.2 Schematic Composite Process for Feature Generation

xmlns:pm = http://classakt.aktors.org:8000/ontologies/process_management.owl#
xmlns:tc = http://classakt.aktors.org:8000/ontologies/TC-ProfileHierarchy.owl#
xmlns:sv = http://www.daml.org/services/owl-s/1.1/Service.owl#

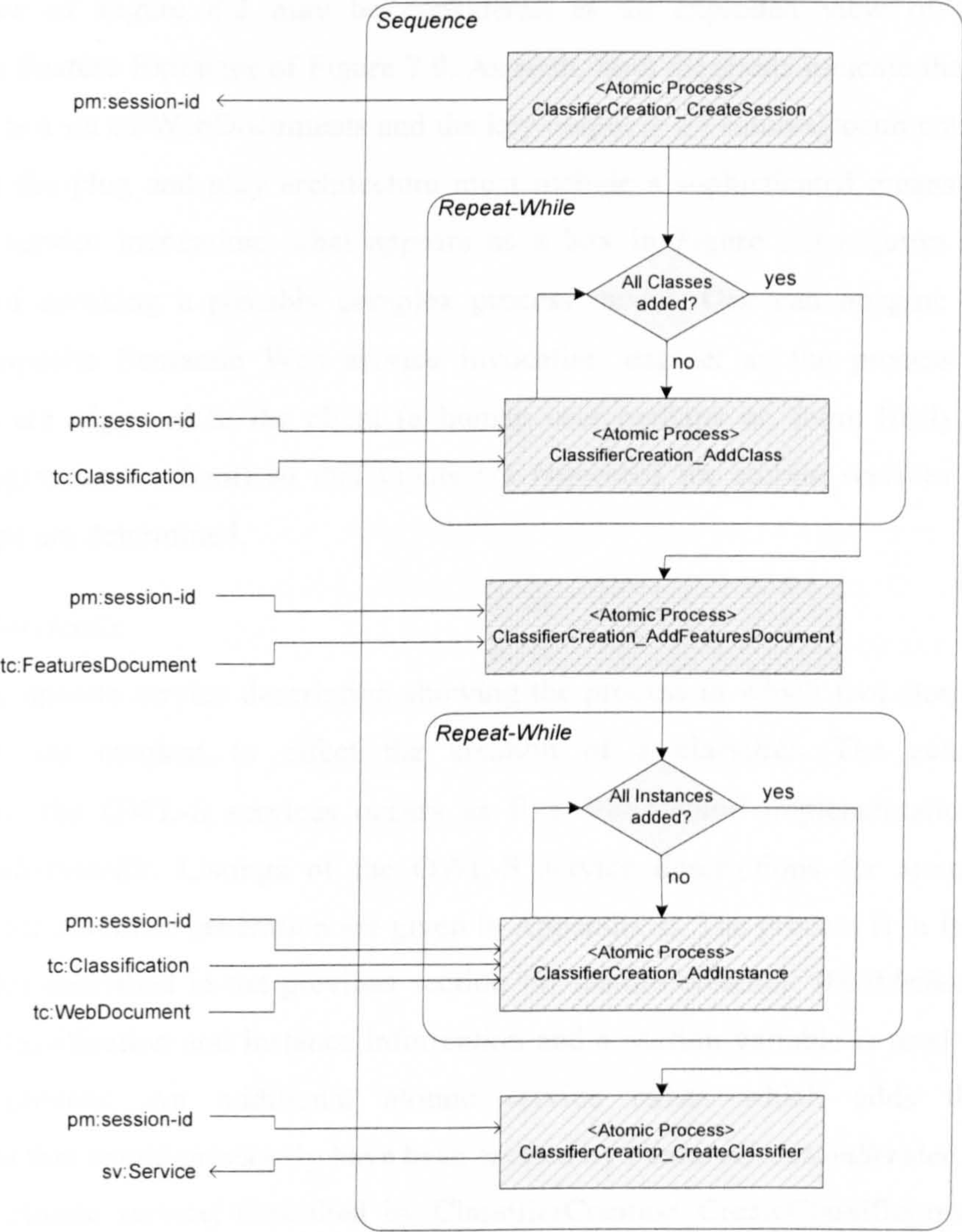


Figure 8.3 Schematic Composite Process for Classifier Service Generation

Section 7.4.3, ‘Pulling Everything Together’ discussed how the careful design of an ontology describing Semantic Web services would facilitate the development of a workbench to permit plug and play between functional components. In particular, the composite service of Figure 8.2 may be considered as an exploded view of the Information Gain Feature Extractor of Figure 7.9. As such, both diagrams indicate that a key set of inputs is a set of WebDocuments and the key output is a FeaturesDocument. It also follows that the plug and play architecture must include a sophisticated means of composite Web service invocation: what appears as a box in Figure 7.9 requires an engine capable of invoking a possibly complex process model. One can imagine an event-driven composite Semantic Web service invocation engine: as the process is executed, events are triggered to the client (a human user perhaps or, more likely, a sophisticated programmatic client) so that inputs are requested for atomic services or conditions of loops are determined.

8.3.3 *ClassifiersServiceEx*

Figure 8.3 is a composite service description showing the process in which five atomic OWL-S services are invoked to effect the creation of a classifier. The actual implementation of the OWL-S services occurs as five *WebMethod* implementations within *ClassifiersServiceEx*. Listings of the OWL-S service descriptions for atomic services required for classifier generation are given in Appendix E. The process is in fact very similar to that described in the previous section for feature creation. *WebMethods* exist for adding classification and instance information and a session variable is used to coordinate the process. An additional atomic service exists which adds the FeaturesDocument that would previously have been created by *FeatureCreationService*.

The final atomic service, described by *ClassifierCreation_CreateClassifier.owl*, actually creates a classifier based on the information supplied to the composite process to date. This new classifier is in fact serialized on the server so that it may be recreated as necessary when the classifier is invoked in the future. A further application, *ClassifierService* has been created which invokes these serialized classifiers and the URL returned as the result of the *ClassifiersServiceEx* process is in fact a URL to an OWL-S service description for *ClassifierService* which invokes the appropriate serialization.

OWL-S service descriptions for new services are themselves generated from the template, `Classifier_ClassifyTemplate.owl`.

Referring back to Figure 7.9, the composite process described here is designed to fit in the plug and play framework and may be considered to be the ‘Naïve Bayes Classifier Generator’ in that figure. At a high level, it has as its input the set of WebDocuments, that is the tagged corpus, and a FeaturesDocument, the set of features generated for the classifier. Its end result is a URL to an OWL-S service description for a new service which classifies Web documents according to the input classification scheme.

8.4 Outstanding Issues

The system described so far in this Chapter is a working system. A Java program has been written which invokes the OWL-S services of the previous two sections: a set of features may be created, and a classifier subsequently generated, which can classify unseen documents according to some classification scheme. The purpose of this section is to note certain outstanding issues which are important to the work carried out here but which have yet to be implemented, primarily due to the lack of available client tools or excessive length of time required to implement them.

8.4.1 Service Advertisement

An implementation of a framework for Semantic Web services requires a component that manages the advertisement and matching of services to their clients. For such a system one can imagine a registry of OWL-S service profiles that may be queried by various clients to find services of use to them. This is the role fulfilled by UDDI [UDDI.org, 2000] currently for conventional services. For OWL-S, an analogue to UDDI does not yet exist although some efforts have been made to address the issue. For example in [Paolucci et al., 2002a, Paolucci et al., 2002b] a matching engine between DAML-S service profiles and their clients is considered together with the possibilities of integrating UDDI into a Semantic Web services framework. Nevertheless, it is important to model the service advertisement as in a usable framework a service is of limited use if it cannot be found.

For the classifiers framework, two important features may be noted which facilitate service advertisement. Firstly, as outlined in section 7.4 ‘An Ontology for Text Classification’, the OWL-S service profile is sub-classed by a *TC-Resource* and then further sub-classed as necessary to distinguish feature extractors, classifier generators, etc. This typing enables searches of a service registry for a particular artefact. For example, if a client requires a classifier generator, then profiles may be filtered by *TC-ClassifierGenerator*.

```
<profile:serviceCategory>
  <profile:ServiceCategory>
    <profile:taxonomy>
      http://classakt.aktors.org:8000/ontologies/digital_archive_acm.owl
    </profile:taxonomy>
    <profile:value>http://www.acm.org/class/1998/C.0
      http://www.acm.org/class/1998/C.1 http://www.acm.org/class/1998/C.2
      http://www.acm.org/class/1998/C.3 http://www.acm.org/class/1998/C.4
      http://www.acm.org/class/1998/C.5</profile:value>
    </profile:ServiceCategory>
  </profile:serviceCategory>
  <profile:serviceParameter>
    <profile:ServiceParameter>
      <profile:serviceParameterName
        rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Breakeven
      </profile:serviceParameterName>
      <profile:sParameter>0.74</profile:sParameter>
    </profile:ServiceParameter>
  </profile:serviceParameter>
```

Figure 8.4 OWL-S Service Advertisement for a Classifier Service

Secondly, the OWL-S service profile is designed to enable the publication of additional information such as quality guarantees that are provided by the service, possible classification of the service and additional parameters that the service may want to specify. This is particularly relevant for the services generated by the classifier generator as these services must advertise what it is they classify. To this end, the OWL-

S profile has the *ServiceCategory* and *ServiceAttribute* attributes. The text in Figure 8.4 is for illustrative purposes and demonstrates the minimum information that a classifier service profile might advertise in order that clients may determine if it is appropriate

Referring to Figure 8.4, use of the *ServiceCategory* class enables a list of classes relevant to the classifier to be enumerated. The *taxonomy* attribute types the relevant class structure and the comma separated list of classes in the *value* attribute is the enumeration of classes. The *ServiceParameter* class is used to state a measure of effectiveness of the classifier for which the breakeven point¹ is, perhaps, the most straightforward measure.

8.4.2 Describing Services in an Ontology

The end product of the composite service of Figure 8.3 is some RDF locating an OWL-S service. While this is satisfactory, there is an argument for developing a high-level ontological view of the OWL world.

So, how can a service be alternatively represented in an ontology? If technologically the Semantic Web is considered a ‘system’, then a service architecture would be considered a core component within this system. Further, it seems natural and useful to consider specifying the system itself: such a specification defines a high-level view of the Semantic Web world and consequently the design of all implemented components (ontologies, services, etc) ought themselves to be rooted in this high level view.

In fact, this is the approach adopted by the Web Service Modeling Ontology (WSMO) community [WSMO, 2004]. WSMO primarily addresses the description of Semantic Web Services and within this context, four key elements are identified which are required to enable a working service environment: ontologies, services, mediators and goals. (Mediators enable interoperability between components using different ontologies whereas goals may be considered representations of an objective for which fulfilment is sought through the execution of a Web service). Indeed, the starting point for the design of WSMO may be considered a high level ontology in which these four core entities are defined, together with an extensive set of properties describing their components. All

¹ See section 4.3

actual components (for example, a classifier in the WSMO world) will then be defined in terms of this ontology.

Such a situation does not arise in the OWL world. OWL is solely a Web ontology language and says nothing about the implementation of a Semantic Web system as such. OWL-S may be considered the most viable service architecture available for OWL. However, research relating to mediators and goals is far from mature with no specific technologies emerging as a default standard and no regard has been given to creating a formal framework incorporating all of these components. The WSMO working group, on the other hand, primarily addresses Semantic Web services and is required to base its design on a working Semantic Web system: hence it has had the opportunity to develop a system, and a system ontology, from the ground up.

So, although an ontology of the system as a whole does not exist for OWL, an initiative in this area may lead to something along the lines shown in Figure 8.5. The OWL-S services may be tied to this ontology by sub-classing the Service class in the OWL-S service ontology with the WebService class.

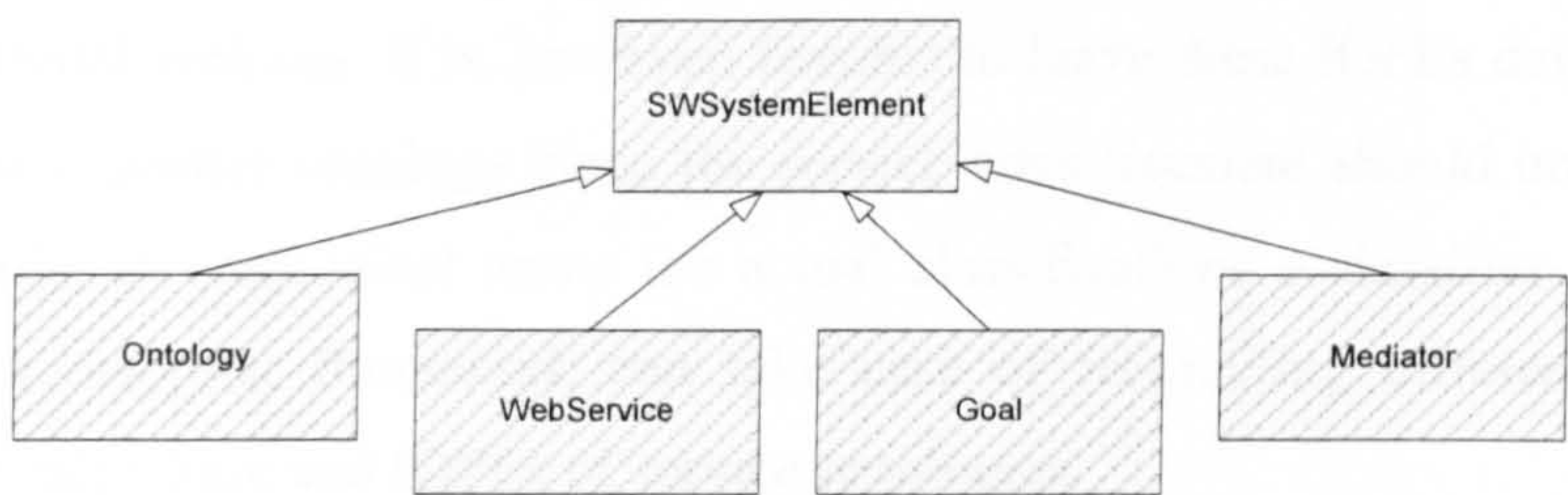


Figure 8.5 Top level System Classes in a Notional OWL Framework

8.5 Ontology Mapping

Chapter 7 presented two ontologies: an ontology for a digital archive derived from an upper ontology (the ABC ontology) and an ontology whose design was driven by the requirement to describe text classifiers within a service environment. The Web service generator presented in this chapter and, in particular, the classifier services it generates have their inputs, outputs, preconditions and effects (IOPEs) defined in terms of the ontology for text classifiers

The Semantic Web will make use of a large number of ontologies and there is a requirement that they are able to interoperate to share metadata and make use of services. This work has hinted at a design pattern in which ontologies for knowledge bases as stores of metadata are designed carefully to make use of generic upper ontologies wherever possible and services are described independently in their own ontology derived from the advertisement component of the service architecture. This suggests that service architectures may be reused by various knowledge bases of which the domain of digital archives is only one, admittedly important, example. The study of *ontology mapping* is an effort to address the interoperation of ontologies. In the context of this work in which the classifier generator is required to be used by agents working within the framework of a digital archive, it is required to overcome two issues:

- The discovery and invocation of a mapping between pertinent IOPEs of the digital archive and text classifier ontologies in order that the (feature generator and) text classifier generator may be invoked.
- To enable the possibility that the IOPEs of OWL-S descriptions of generated services may be defined in terms of the client ontology, in this case the ontology for digital archives. It is, however, feasible to leave these IOPEs defined in terms of the classifier ontology since the service advertisement should unambiguously describe in ontological terms the actual classifications a classifier knows about and a mapping framework can take care of reconciling differences between knowledge base and IOPEs of service ontologies.

Much research has been invested into the field of ontology mapping and a comprehensive survey of the various techniques is available in [Kalfoglou & Schorlemmer, 2003]. Of particular interest (and as a particular solution which may be taken to suggest a practical solution to the issue of mapping in connection with this work) is a framework, called MAFRA – A Mapping FRamework for Distributed Ontologies, introduced by a group from the University of Karlsruhe [Maedche et al., 2002].

MAFRA first introduces a conceptual framework which describes the steps to be undertaken to map two ontologies and then to populate instances from one ontology into instances of the second. This is shown in Figure 8.6. The horizontal steps focus on the key stages in the process. *Lift & Normalization* refers to raising all data onto the same

representation level, in particular RDF. *Similarity* refers to various algorithms and techniques which establish similarities between entities in the two ontologies. As a simple example, lexical similarity may be sought between concepts. *Semantic Bridging* establishes the mapping between the two ontologies. This mapping is formalised in terms of MAFRA’s Semantic Bridging Ontology (SBO) which is discussed later on. There are five steps in semantic bridging. First and second, concepts and properties are bridged. Third, an inferencing step maps concepts that do not have specific counterparts in both ontologies. For example, an EMPLOYEE in one ontology may be mapped to an ACADEMIC in a second ontology. Fourth, a refinement step intends to improve quality of bridges between a source concept and the sub-concepts of target concepts. Fifth, and largely dependent on a domain expert, a transformation specification step associates transformation procedures to the translation. For example, a mapping may be required between different types of properties.

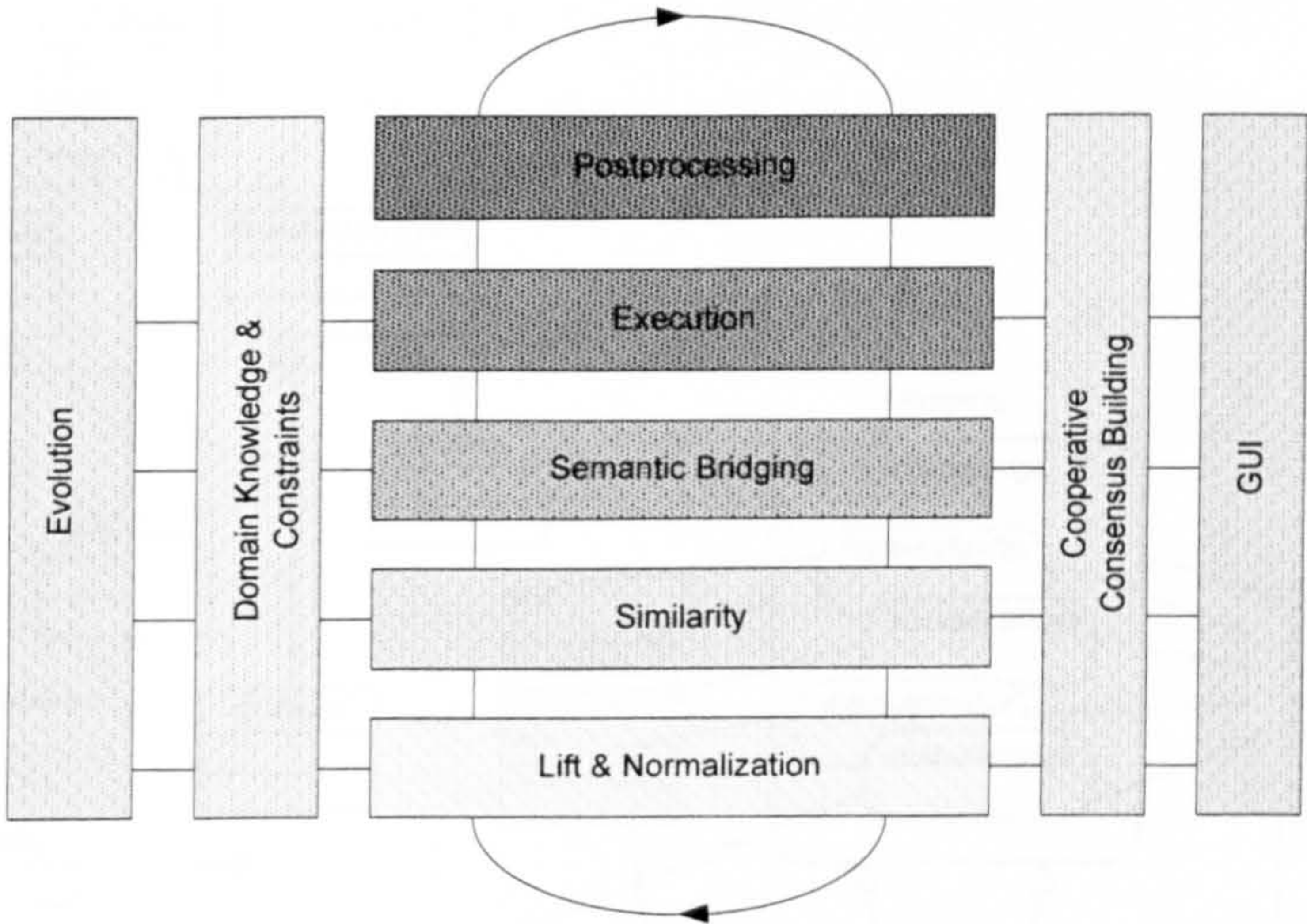


Figure 8.6 MAFRA Conceptual Architecture
(adapted from [Maedche et al., 2002])

The vertical dimensions of the process are modules which are valid throughout the mapping process. *Evolution* refers to processes in place which enable the mapping to be maintained when the ontologies themselves are revised. *Cooperative Consensus*

Building takes account of establishing a consensus on semantic bridges between the two communities participating in the mapping process. Introduction of *Domain Constraints* and *Background Knowledge*, for example glossaries or lexical ontologies, can improve the quality of similarity computation and semantic bridging. A *Graphical User Interface* (GUI) is key to assisting human input into the process.

The key innovation of the MAFRA framework, however, is its Semantic Bridging Ontology (SBO): ‘The role of a semantic bridge is to encapsulate all necessary information to transform instances of one source ontology entity to instances of one target ontology entity’. The SBO is itself an ontology and mappings are defined in terms of it, much as service descriptions are described in terms of OWL-S. Comprehensive details are available in [Maedche et al., 2002]. However, its essential function is most conveniently illustrated by means of an example and Figure 8.7 is such an example drawn as a UML diagram copied from this same paper.

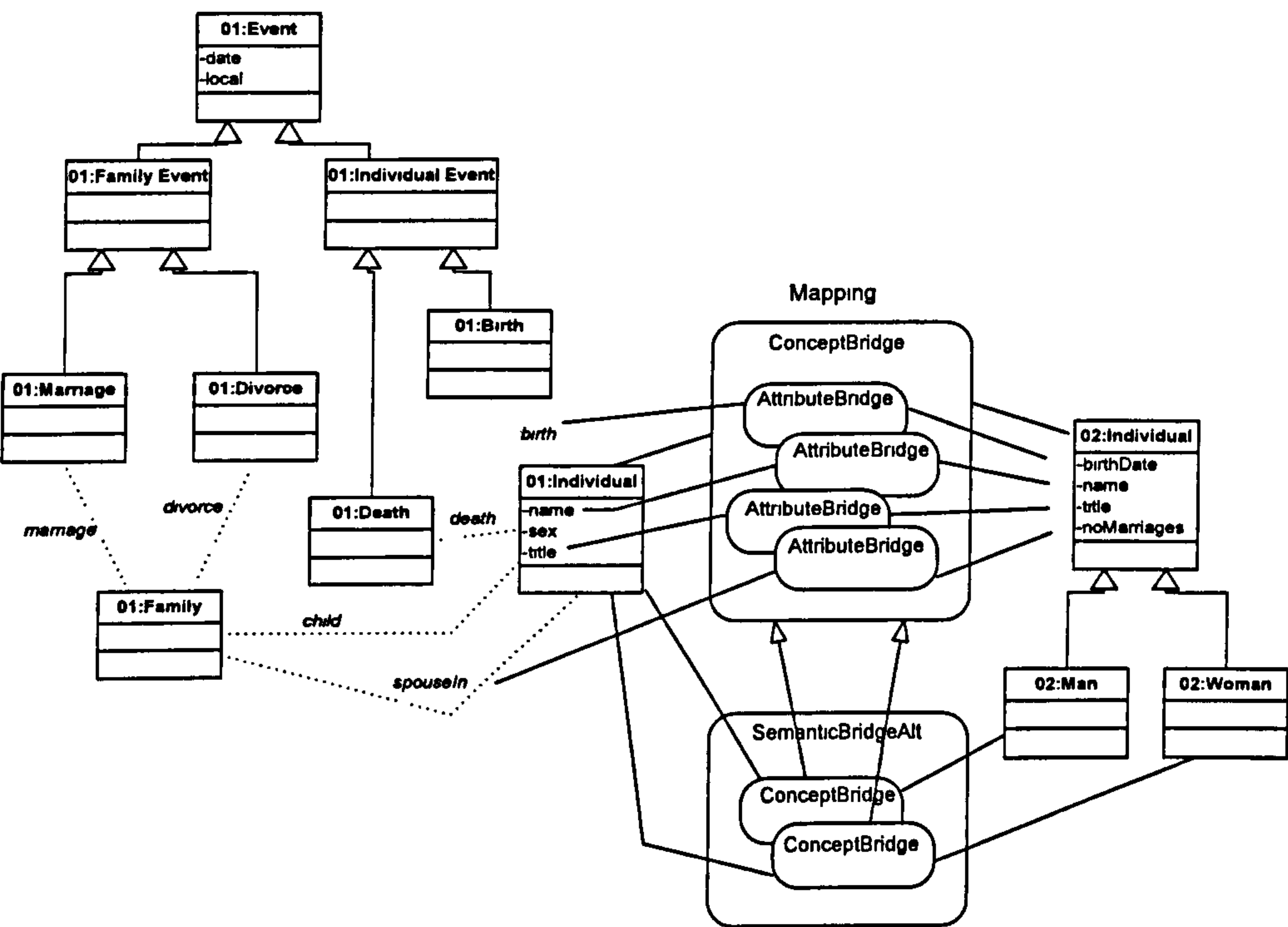


Figure 8.7 MAFRA Mapping of Two Ontologies
(adapted from [Maedche et al., 2002])

The two ontologies of Figure 8.7 describe information about people. On the left hand side, ontology 01 is designed to portray knowledge about royal families and shows information such as births, deaths and marriages. On the right, ontology 02 is a less complex ontology with a simpler representation of people. The centre of the diagram illustrates the MAFRA mapping extracted from its DAML+OIL serialization. Central to this mapping is the Individual-Individual concept bridge. In 02, instances are created of 02:Man and 02:Woman, such that, in object-oriented terms, 02:Individual is abstract. Consequently, the Individual-Individual concept bridge is abstract. Instead, the SemanticBridgeAlt construct is utilised to provide concrete mappings between individuals in the two ontologies. The two ConceptBridges of the SemanticBridgeAlt construct define these concrete mappings and are enumerated with additional conditional information to assert the gender mappings across the ontologies. The Individual-Individual ConceptBridge is the appropriate placeholder for the mapping of the two sets of properties. Such mappings are typically associated with a service to enable the property translation. For example, when translating the 'name' property, a service to simply copy strings may be associated whereas when calculating 02:noMarriages, a service is required which counts the spouseIn attributes for an instance published in ontology 01.

xmlns.dav = http://classakt.aktors.org:8000/ontologies/digital_archive.owl#
xmlns.tc = http://classakt.aktors.org:8000/ontologies/TC-ProfileHierarchy.owl#

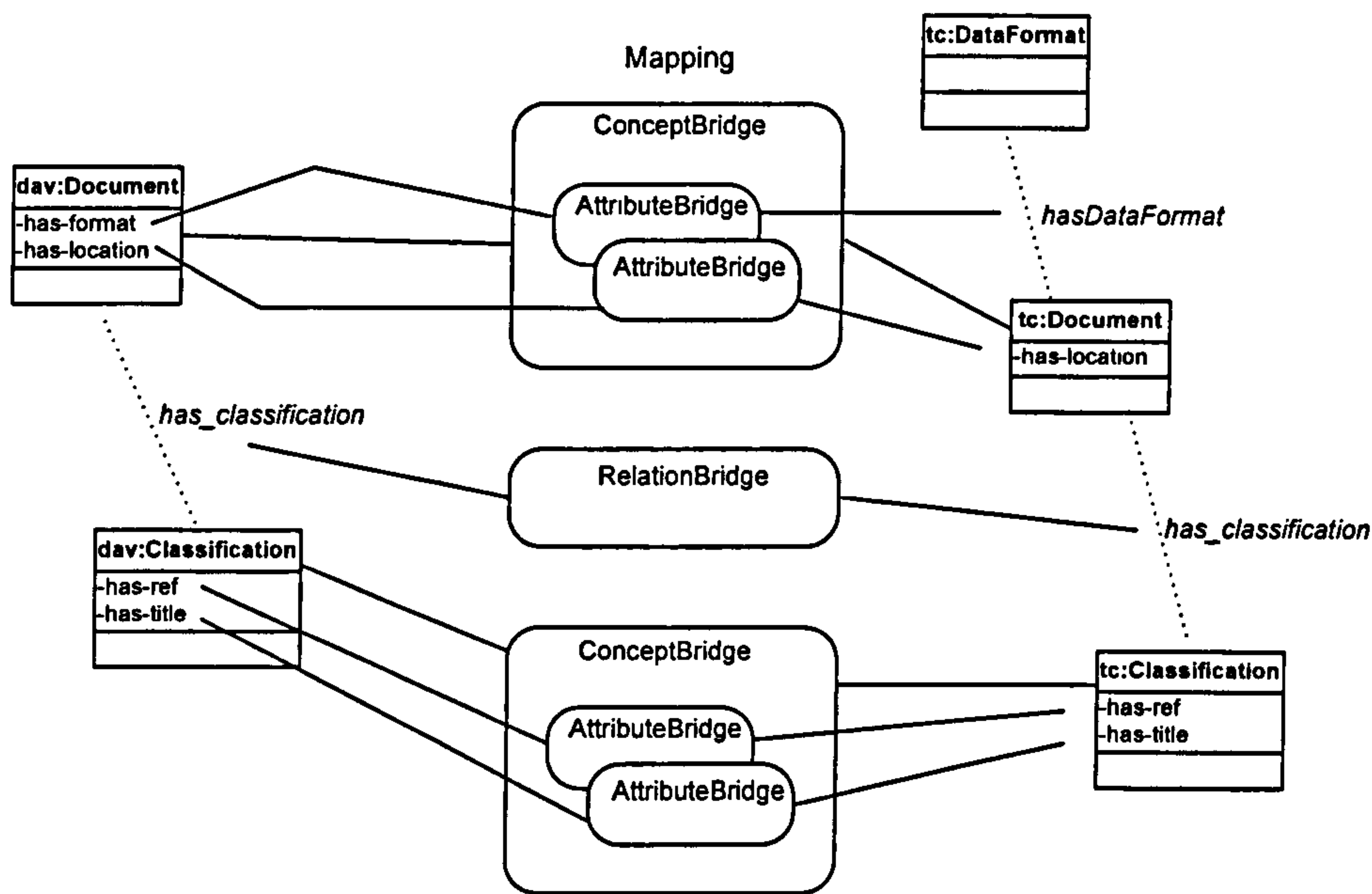


Figure 8.8 MAFRA Mapping of Text Classification and Digital Archive Ontologies

By contrast, the mapping between the document and classification representations of the digital archive and the text classification ontologies of this thesis is as simple as a mapping is likely to get. This is illustrated in Figure 8.8. Since the designer of the ontologies is the same, the designs are almost the same.

A framework for mediation such as MAFRA may be considered as a candidate for a formal framework of mediation in the OWL/OWL-S world. Such a mapping ontology is amenable to an infrastructure which at its core has a processing component which is able to coordinate the mapping process between various ontologies. Such an infrastructure might consist essentially of a framework such as that detailed in Figure 8.6. In relation to this work, such a processing component may be a service that is required:

1. To be a repository for mappings and to be well advertised as such.
2. When supplied with a target ontology or fragment thereof, to return a condition stating the extent to which a mapping exists to a target ontology. (Simply put, whether it is complete, partial or non existent). In particular, such a service may be capable of returning the intersection or difference of the source ontology.

3. When supplied with some instance knowledge, in RDF or OWL, from the source ontology, translate it into instance knowledge of the target ontology.

Given the existence of such a framework, it could be invoked by *FeatureCreationService* and *ClassifiersServiceEx* in two contexts:

- As part of the service invocation process. A client in the digital archive world searches for, and then invokes, a set of services capable of creating an actual text classifier. Part of finding such a set of services includes evaluating whether a suitable mapping exists or else providing a framework to create one. On invoking the services, the mapping process from instances in the digital archive ontology to the service IOPEs is transparently made, utilising the mapping framework.
- The client in the digital archive world may require that the IOPEs of the classifier service generated are made in terms of concepts/attributes in the digital archive world. To enable this, an additional input to the *ClassifierCreation_CreateSession* atomic service can be considered which is the (digital archive) ontology. This service may query the mapping framework to evaluate appropriate IOPE specifications in the final OWL-S classifier service description.

This section has explored a key requirement of making not just a Web service generator, but all services, work on the Semantic Web, namely the interoperation of ontologies. As the work in this thesis has demonstrated, the technology is becoming available to make the development of Semantic Web based applications both interesting and viable. Certainly, this introduction to a mapping procedure is a very broad portrait which raises a number of questions and implementation issues which it would be interesting to pursue. A particularly interesting question is to what extent can ontology mapping be made a seamless and painless procedure to use? The ontology mapping literature focuses on various techniques to map ontologies which are independently designed and consequently likely to be very different. As the example from MAFRA in Figure 8.7 shows, mapping even two small fragments of an ontology can be a relatively complex process requiring the intervention of an expert in both the subject domain and mapping process.

Although the mapping for classifiers in Figure 8.8 is fairly straightforward, it is perhaps revealing of a constructive path forward. It would be interesting to pursue the

possibility of ‘reusable ontological fragments’. These fragments would be generic, well designed ontological representations of useful concepts which could be stored in a library which might then be searched and queried by ontology designers. As a relevant example, there is the design of a document that was mapped in a straightforward way in Figure 8.8. Let us consider, the designer of an ontology of Web search engines. This designer will require a design for a document. If he makes use of a popular ontological fragment published in a well known repository, his design effort becomes easier but in particular, he will be able to seamlessly take advantage of the classifier services because a mapping would have been automatically created between his and the text classifier ontology.

An additional benefit of such fragments is that they can be a ‘hub’ for mappings. If the design of a fragment becomes popular and n ontologies map to it, then mapping between two arbitrary ontologies can take place through the hub so that only n mappings are required. Without such a formalism, a mapping may be required between all the constituent ontologies, that is $n(n-1)/2$ mappings each of which would have to be individually considered. Although this scenario may be considered hypothetical, it is certainly an example of the type of problem that an implemented system, such as that explored in this thesis, is well placed to explore.

8.6 Deployment Issues

Performance is an important topic to be addressed. The performance graphs of Chapter 4 were produced on a standard PC (2.40 GHz, 1.0 GB of RAM). Many of the classifiers can be built in reasonable amounts of time and, in particular, Naïve Bayes classifiers may be constructed in under five minutes. However, to build a SVM classifier for the ACM corpus classes A-K using 6000 files takes over six hours. In this case, the eleven classes require the construction of a committee of 45 binary classifiers.

This potential performance bottleneck impinges up on both types of users described previously in this chapter. Semantic Web developers would typically not expect or be prepared to wait hours for their classifier to be built and they may require a large number of such classifiers if they are establishing a service such as ClassAKT which coordinates a hierarchy of classifiers. Researchers, though perhaps more considerate of the time factors involved in building classifiers, are nevertheless likely to

make many more builds of classifiers as a result of their experimentation. Any improvement in performance is likely to be viewed as a very positive development.

The service-oriented architecture described in this chapter would at first seem only to exacerbate the problem. Introducing a network would seemingly entail the added overhead of additional network communication and, in particular, the problem of having to upload whole corpora to a server in order that they may be subjected to classification. In practice, one would expect that servers would cache corpora so that only metadata about the corpus to be examined would be transmitted. In addition, a simple and necessary improvement to the process models of Figures 8.2 and 8.3 is to be able to upload all metadata, for example that pertaining to WebDocuments, in one function call thus rendering the Repeat-Until loops of those Figures a useful, but not really necessary, feature of the processes.

However, this same service-oriented architecture prompts the most viable solution for overcoming the performance bottleneck. This is because a service-oriented architecture is precisely what is required by the Semantic Grid. The Semantic Grid [De Roure et al., 2005, Goble & De Roure, 2004] is an effort to apply Semantic Web technology to the Grid. The Grid is an effort to bring computing resources together, and in particular, in connection with this work, it assembles a large scale data processing capability. The processing back end of a classifier service generator can in fact be tied to a network of high performance computers to speed performance. In this way, by being able to connect to parallel computing resources, classifiers such as SVM which consist of a number of binary classifiers can be created far more quickly (in creation of a number of classifiers, each classifier can be constructed completely independently of the other classifiers). If we build 45 binary classifiers on 45 processors the whole classifier construction process is 45 times quicker.

The Semantic Grid is currently an area of very active research and development. It is an early adopter and advocate of Semantic Web technology and in conjunction with the work in this thesis, provides the opportunity to establish classifier technology as a more practically viable and, therefore useful, set of functionality within the framework of the Semantic Web.

8.7 Summary

By publishing knowledge in a machine processable way, the Semantic Web will improve the user experience in many ways, for example, enabling the visualization of additional information and providing new techniques to search and filter this knowledge. In addition, the Semantic Web provides the solid foundations for a service framework which can assist this process in many ways. This chapter has explored this possibility by not only considering the usefulness of a service to automatically generate metadata, but in particular describing a novel Web service generator which generates such services themselves for a particular domain - in this case for text classifiers. Such frameworks are useful as they abstract problems solved by individual services to provide reusable frameworks which solve as wide a range of a particular problem space as is possible. A problem is solved once and by situating it in the Semantic Web once, it reduces the complexity of an already highly complex and technical environment. In addition, other issues have been addressed such as service advertisement and ontology mapping. By pinpointing the advances so far made, and the shortcomings of the current technology, it becomes possible to shed insights on a future holistic framework for the Semantic Web. Given the essential framework, it is surely through actual implementation of problem domains, and solving the problems that arise, that the Semantic Web will mature and become a reality.

Chapter 9 Conclusions and Future Work

9.1 Conclusions

This thesis began with the description of a problem domain encountered within an existing research project. That is, within the AKT project there was a requirement to extend classification metadata to unseen documents, to enable effective automatic classification of Computer Science related documents to facilitate the addition of these documents to the AKT Triplestore repository. Thus, at a high level, this thesis might be described as an effort to investigate the issues to apply machine learning based text classification technologies within the context of the Semantic Web. Consequently, this work has progressed along two key lines of enquiry:

1. To investigate the current state of the art in classifier technology and the additional requirements necessary to mould this technology to a services framework.
2. To pursue the current standards and approaches pertaining to the Semantic Web with the objective of making classifier technology useful and available on the Semantic Web.

9.1.1 Introduce Text Classifier Technology to the Semantic Web

The taxonomy presented by the ACM, which has been incorporated into the AKT Reference Ontology, describes Computer-Science-related scientific papers. In particular it has been used to manually classify the repository of documents published by the ACM.

This thesis has, for the first time, published results of automatic classification of this data using two high-quality text classifiers: Naïve Bayes and more particularly, Support Vector Machines. It has been found that while results are reasonable (according to measures of precision and recall), they are not excellent: They are perhaps more suited to a recommender system than an automatic classifier. This contrasts with the results for another corpus, Reuters-21578, already well studied in the literature and which has different textual characteristics that enable classifiers trained on it to perform much more effectively.

From these results it is argued that classifier research has, to date, concerned itself primarily with investigating new algorithms on a limited range of test corpora whereas, to be a worthwhile technology on the Semantic Web, exhaustive research is required on a wide variety of corpora using a number of different parameters of which corpus size and feature number are just two significant examples. From results taken this way, a set of heuristics may be developed which enable an effective classifier to be constructed based simply on the parameters of a corpus. On the other hand, it might be found that there are characteristics of a corpus which make it poorly suited to current approaches thereby yielding a rich vein of research in an effort to overcome this issue. Certainly, this would be the initial conclusion from the results published here for the ACM corpus.

In an effort to underline the methodology argued for in this work, results have been published as pairs of graphs for a given corpus and classifier, and for varying corpus size and feature number. These results are an initial effort to show how heuristics may be derived for different sets of textual and classifier parameters.

9.1.2 A Semantic Web Application

Semantic Web infrastructure can now be considered to be at the stage where the fundamental technologies are in place and it becomes interesting to consider both what is possible and what are the shortcomings in respect of building useful applications on the Web. Certainly the seamless integration of the Semantic Web into today's Web is fundamentally what Tim Berners-Lee's vision is about and if such a vision is to become a reality, then the essential technologies should be readily comprehensible and straightforward enough to be understood by an average Web developer. Consequently

Semantic Web applications ought to be visible through the fundamental technology available today.

Through the valuable requirement of text classification, this thesis has sought to put a useful Semantic Web application on the table and discuss issues associated with it. Text classifiers themselves require a service framework and are associated with an ontology which describes both the classification service components of such a system and the specification of a document (and its classifications). Such a service framework might be considered as significantly under achieving its potential for usefulness if it cannot be applied to new domains. In particular, the domain of digital archives is introduced as a well-defined application area which requires classifier services. (A second application area is exemplified by the AKT Reference Ontology). To make such a system feasible, the importance of an ontology mapping infrastructure was highlighted.

The real value of this effort lies in the building of a real system but also in considering it as a vision of the Semantic Web in microcosm. By initiating a feedback loop - in which a working application may be compared and contrasted with a vision of a Semantic Web based on generic technology - the Semantic Web vision becomes one step closer to a reality.

9.1.3 Novel Approach

Creating one-off classifier services such as ClassAKT is fine as far as it goes. However, classifier technology is far more useful if services can be created on the fly after being given various arbitrary corpora. This work has introduced Web service generation. That is, employing a service to generate arbitrary services, as the solution to the problem. It is argued that a Web service generator specialising in the generation of classifier services is required to be constructed so that it contains a set of classifier heuristics (see 9.1.1). Then, any Web developer with no particular knowledge of classifier technology can supply his own tagged corpora to such a generator and derive his own specialised classifier services. This design pattern may be abstracted to a generic solution which will be widely applicable to a range of service application domains on the Semantic Web.

9.2 Future Work

At a high level, future work leading from this thesis may even be the implementation of the Semantic Web itself! Earlier, it was pointed out that, from an OWL perspective, a key missing element is what effectively amounts to an ontology of the high-level components in the Semantic Web. Drawn from a WSMO perspective these may be considered to consist primarily of four elements: ontologies, Web services, mediators and goals. Of these elements, the first three are required in a mature form to implement the system described in this thesis and it is these three that are considered to be required to define a useful system. In turn, OWL has achieved some maturity as an ontology language achieving status as a W3C recommendation. OWL-S is now usable via an API in the form of OWL-S API and a framework such as MAFRA [Maedche et al., 2002] could relatively easily be implemented to provide a functioning mediator framework.

With this in mind, the application framework in this thesis, that is the domain of digital archives interacting with the service architecture of text classifiers, could be considered an ideal domain to showcase working Semantic Web technology on the ‘common Web’. Document archives are a useful and prevalent resource very amenable to development into knowledge bases. Indeed, as previously mentioned, a project called SIMILE¹ is already underway to interface the digital archive DSpace² to the Semantic Web. Document classification is also a technology of high importance on the Semantic Web. After all, the Web essentially consists of text based documents and basic metadata about these documents is classification metadata.

However, correct implementation of this application framework still poses a number of technological problems not least of which is a requirement for careful design: so that experts from various fields work together to produce an elegant solution. In particular the following issues need to be addressed with the aim of producing de facto standards in the field

- Text classification research is required to establish heuristics for the classification of corpora according to key textual parameters and standard machine learning algorithms. Publishing such information will specify the boundaries of what text

¹ See simile.mit.edu

² See dspace.org

can be effectively automatically classified. This requirement is pressing due to the importance of classification metadata, particularly in connection with the Semantic Web. (Of course, such information may alternatively be derived privately within the confines of a company).

- Consensus is required for the sound design of an ontology for archives and in particular, digital archives. As this work has pointed out, this is not simply a question of a number of ontologists and experts in archives sitting down and writing an ontology. Such an ontology must be created from the perspective of the wider Semantic Web and key domains and service architectures, in particular the text classifier architecture, must be designed at the same time to facilitate their mapping and integration. Indeed, this thesis has argued that reusable ontological components (for example, a 'document' component) should be designed such that they can be plugged into more general ontologies such as that for digital archives. To continue with, and to further implement, the system introduced in this thesis, three ontological 'systems' need to be considered and built up at the same time: an ontology for text classifiers, an ontology for (digital) archives and ontological components which model the same entities between the two ontologies or which are generic to the extent that they may be required by other ontologies.
- An ontology of core Semantic Web system components needs to be derived, as mentioned previously. This will not only define these key system components (most likely ontologies, Web services, mediators and goals) but also will focus the collective mind of the community to define standards where they are lacking. In particular for this work, establishing a mediator framework in the OWL world.

Appendix A ClassAKT Service Ontologies

A.1 Guidance Notes

The ontologies listed in this appendix together comprise the OWL-S Semantic Web service interface to ClassAKT, the atomic Semantic Web classifier service introduced in Chapter 5. As specified by OWL-S, ontologies have been derived for the service profile, process and grounding together with a service ontology (Appendix A.2) that binds them together. In addition, a WSDL interface is listed in Appendix A.6.

The ontologies for ClassAKT were created by the author in a basic text editor using, for guidance, example OWL-S ontologies published by the OWL-S consortium. At the time of authoring, tools that consumed OWL-S service descriptions were not available so the ontologies listed here serve to sketch the principles behind OWL-S rather than providing a completely correct, working interface to a Semantic Web service.

A.2 Service Ontology

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://classakt.actors.org/services/owl-s/0.9/Service.owl">
  <!ENTITY ca_profile "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTProfile.owl">
  <!ENTITY ca_process "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTProcess.owl">
  <!ENTITY ca_grounding "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTGrounding.owl">
```

```

    <!ENTITY DEFAULT "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTService.owl">
]>

<rdf:RDF
  xmlns:rdf  = "&rdf;#"
  xmlns:rdfs = "&rdfs;#"
  xmlns:owl  = "&owl;#"
  xmlns:xsd  = "&xsd;#"
  xmlns:service= "&service;#"
  xmlns      = "&DEFAULT;#"
>

  <owl:Ontology>
    <owl:versionInfo>
      $Id: ClassAKTService.owl,v 1.00 2003/09/30 16:00:00 steve Exp $
    </owl:versionInfo>
    <rdfs:comment>
      This ontology represents the OWL-S service description for the
      ClassAKT web service example.
    </rdfs:comment>
    <owl:imports rdf:resource="&owl;"/>
    <owl:imports rdf:resource="&service;" />
    <owl:imports rdf:resource="&ca_profile;" />
    <owl:imports rdf:resource="&ca_process;" />
  </owl:Ontology>

  <service:Service rdf:ID="ClassAKT">

    <!-- Reference to the ClassAKT Profile -->
    <service:presents rdf:resource="&ca_profile;#ClassAKT_Profile"/>

    <!-- Reference to the ClassAKT Process Model -->
    <service:describedBy rdf:resource="&ca_process;#ClassAKT_ProcessModel"/>

    <!-- Reference to the ClassAKT Grounding -->
    <service:supports rdf:resource="&ca_grounding;#Grounding_ClassAKT"/>

  </service:Service>

</rdf:RDF>

```

A.3 Service Profile

```

<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY service "http://www.daml.org/services/owl-s/0.9/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/0.9/Profile.owl">
  <!ENTITY profileHierarchy "http://www.daml.org/services/owl-
s/0.9/ProfileHierarchy.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/0.9/Process.owl">
  <!ENTITY ca_service "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTService.owl">
  <!ENTITY ca_process "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTProcess.owl">
  <!ENTITY country "http://www.daml.org/services/owl-s/0.9/Country.owl">
  <!ENTITY concepts "http://www.actors.org/ontology/portal">
  <!ENTITY DEFAULT "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTProfile.owl">

```

```

]>

<!--
This document uses entity types as a shorthand for URIs.
Download the source for a version with unexpanded entities.
-->

<rdf:RDF
  xmlns:rdf=      "&rdf;#"
  xmlns:rdfs=     "&rdfs;#"
  xmlns:owl =     "&owl;#"
  xmlns:service=  "&service;#"
  xmlns:process=  "&process;#"
  xmlns:profile=  "&profile;#"
  xmlns:profileHierarchy= "&profileHierarchy;#"
  xmlns=         "&DEFAULT;#">

  <owl:Ontology>
    <owl:versionInfo>
      $Id: ClassAKTProfile.owl,v 1.13 2003/10/01 15:00:00 steve Exp $
    </owl:versionInfo>
    <rdfs:comment>
      ClassAKT Example for OWL-S release Profile description
    </rdfs:comment>

    <owl:imports rdf:resource="&rdf;" />
    <owl:imports rdf:resource="&rdfs;" />
    <owl:imports rdf:resource="&owl;" />
    <owl:imports rdf:resource="&service;" />
    <owl:imports rdf:resource="&profile;" />
    <owl:imports rdf:resource="&process;" />
    <owl:imports rdf:resource="&ca_service;" />
    <owl:imports rdf:resource="&ca_process;" />
    <owl:imports rdf:resource="&concepts;" />
    <owl:imports rdf:resource="&country;" />
    <owl:imports rdf:resource="&profileHierarchy;" />
  </owl:Ontology>

  <!-- ##### -->
  <!-- # Instance Definition of ClassAKT Advertisement # -->
  <!-- ##### -->

  <!--

    ClassAKT is a service which classifies computing related publications
    according to the ACM Computing Classification System

    -->

  <profileHierarchy:InformationSource rdf:ID="ClassAKT_Profile">
    <!-- reference to the service specification -->
    <service:presentedBy rdf:resource="&ca_service;#ClassAKT"/>
    <!-- reference to the process model specification -->
    <profile:has_process rdf:resource="&ca_process;#ClassAKT_ProcessModel"/>

    <profile:serviceName>ClassAKT</profile:serviceName>
    <profile:textDescription>
      ClassAKT is a service which classifies computing related publications
      according to the ACM Computing Classification System
    </profile:textDescription>

```

```

<!-- specification of contact information -->

<!-- other contact -->
<profile:contactInformation>
  <profile:Actor rdf:ID="ClassAKT-author">
    <profile:name>Stephen Ball</profile:name>
    <profile:title>Service author</profile:title>
    <profile:email>swb02r@ecs.soton.ac.uk</profile:email>
    <profile:physicalAddress>
      School of Electronics and Computer Science,
      University of Southampton,
      SO17 1BJ,
      United Kingdom
    </profile:physicalAddress>
    <profile:webURL>
      http://www.ecs.soton.ac.uk/info/#person-04929
    </profile:webURL>
  </profile:Actor>
</profile:contactInformation>

<!-- Descriptions of IOPEs -->

<profile:input>
  <profile:ParameterDescription rdf:ID="ComputingPublication">
    <profile:parameterName> ComputingPublication </profile:parameterName>
    <profile:restrictedTo rdf:resource="& concepts;#Publication"/>
    <profile:refersTo rdf:resource="& ca_process;#url_In"/>
  </profile:ParameterDescription>
</profile:input>

<profile:output>
  <profile:ParameterDescription rdf:ID="Classification">
    <profile:parameterName> Classification </profile:parameterName>
    <profile:restrictedTo rdf:resource="& concepts;#Generic-Area-Of-
Interest"/>
    <profile:refersTo rdf:resource="& ca_process;#classes_Out"/>
  </profile:ParameterDescription>
</profile:output>

</profileHierarchy:InformationSource>
</rdf:RDF>

```

A.4 Service Process

```

<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/0.9/Service.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/0.9/Process.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/0.9/Profile.owl">
  <!ENTITY ca_service "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTService.owl">
  <!ENTITY concepts "http://www.actors.org/ontology/portal">
  <!ENTITY DEFAULT "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTProcess.owl">
]>

```

```

<rdf:RDF
  xmlns:rdf=      "&rdf;#"
  xmlns:rdfs=     "&rdfs;#"
  xmlns:owl =    "&owl;#"
  xmlns:xsd=      "&xsd;#"
  xmlns:service=  "&service;#"
  xmlns:process=  "&process;#"
  xmlns:profile=  "&profile;#"
  xmlns=          "&DEFAULT;#">

  <owl:Ontology rdf:about="">
    <owl:versionInfo>
      $Id: ClassAKTProcess.owl,v 1.00 2003/10/01 15:00:00 steve Exp $
    </owl:versionInfo>
    <rdfs:comment>
      ClassAKT Example for OWL-S release Process Model
    </rdfs:comment>

    <owl:imports rdf:resource="&rdf;" />
    <owl:imports rdf:resource="&rdfs;" />
    <owl:imports rdf:resource="&owl;" />
    <owl:imports rdf:resource="&service;" />
    <owl:imports rdf:resource="&process;" />
    <owl:imports rdf:resource="&profile;" />
    <owl:imports rdf:resource="&concepts;" />
  </owl:Ontology>

  <!-- ##### -->
  <!-- Instance Definition of ClassAKT Process Model -->

  <process:ProcessModel rdf:ID="ClassAKT_ProcessModel">
    <process:hasProcess rdf:resource="#ClassAKT_Process" />
    <service:describes rdf:resource="&ca_service;#ClassAKT"/>
  </process:ProcessModel>

  <!--
##### -->
  <!-- ClassAKT (ATOMIC)
    Get ACM classification for some scientific document pertaining to
computing
  -->

  <owl:Class rdf:ID="ClassAKT">
    <owl:subClassOf rdf:resource="&process;#AtomicProcess" />
  </owl:Class>

  <!-- ClassAKT IOPEs -->

  <rdf:Property rdf:ID="url_In">
    <owl:subPropertyOf rdf:resource="&process;#input"/>
    <owl:domain rdf:resource="#ClassAKT"/>
    <owl:range rdf:resource="&concepts;#Publication"/>
  </rdf:Property>

  <rdf:Property rdf:ID="classes_Out">
    <owl:subPropertyOf rdf:resource="&process;#output"/>
    <owl:domain rdf:resource="#ClassAKT"/>
    <owl:range rdf:resource="&concepts;#Generic-Area-Of-Interest"/>
  </rdf:Property>

</rdf:RDF>

```

A.5 Service Grounding

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf          "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs         "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl         "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd           "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service      "http://www.daml.org/services/owl-s/0.9/Service.owl">
  <!ENTITY grounding    "http://www.daml.org/services/owl-
s/0.9/Grounding.owl">
  <!ENTITY ca_service   "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTService.owl">
  <!ENTITY ca_process   "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTProcess.owl">
  <!ENTITY ClassAKTGroundingWSDL "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTGrounding.wsdl">
  <!ENTITY DEFAULT      "http://classakt.actors.org/services/owl-
s/0.9/ClassAKTGrounding.owl">
]>

<!--
This document uses entity types as a shorthand for URIs.
Download the source for a version with unexpanded entities.
-->

<rdf:RDF
  xmlns:rdf=          "&rdf;#"
  xmlns:rdfs=         "&rdfs;#"
  xmlns:owl=          "&owl;#"
  xmlns:xsd=          "&xsd;#"
  xmlns:service=      "&service;#"
  xmlns:grounding=    "&grounding;#"
  xmlns=              "&DEFAULT;#">

  <owl:Ontology rdf:about="">
    <owl:versionInfo>
      $Id: ClassAKTGrounding.owl,v 1.00 2003/10/01 15:00:00 steve Exp $
    </owl:versionInfo>

    <rdfs:comment>
      ClassAKT example owl-s description
    </rdfs:comment>

    <owl:imports rdf:resource="&rdf;" />
    <owl:imports rdf:resource="&rdfs;" />
    <owl:imports rdf:resource="&owl;" />
    <owl:imports rdf:resource="&service;" />
    <owl:imports rdf:resource="&grounding;" />
    <owl:imports rdf:resource="&ca_service;" />
    <owl:imports rdf:resource="&ca_process;" />
  </owl:Ontology>

  <!-- ##### -->
  <!-- # Instance Definition of ClassAKT Grounding # -->
  <!-- ##### -->

  <!-- ##### -->
  <!-- # service:ServiceGrounding # -->
```

```

<grounding:Wsd1Grounding rdf:ID="Grounding_ClassAKT">
  <service:supportedBy rdf:resource="&ca_service;#ClassAKT"/>
  <!-- Collecton of all the groundings specifications -->
  <grounding:hasAtomicProcessGrounding
rdf:resource="#Wsd1Grounding_ClassAKT"/>
</grounding:Wsd1Grounding>

<!-- ##### -->
<!-- # ClassAKT
      # -->

<grounding:Wsd1AtomicProcessGrounding rdf:ID="Wsd1Grounding_ClassAKT">
  <!-- Grounding for the Atomic Process ClassAKT -->
  <grounding:daml1Process rdf:resource="&ca_process;#ClassAKT"/>
  <!-- Reference to the corresponding WSDL operation -->
  <grounding:wsdlOperation rdf:resource="#ClassAKT_Operation"/>

  <!-- Reference to the WSDL input message -->
  <grounding:wsdlInputMessage>
    <xsd:anyURI rdf:value="&ClassAKTGroundingWSDL;#ClassAKT_Query"/>
  </grounding:wsdlInputMessage>

  <!-- Mapping of OWL-S inputs to WSDL message parts -->
  <grounding:wsdlInputs rdf:parseType="Collection">
    <grounding:Wsd1InputMessageMap>
      <grounding:daml1Parameter rdf:resource="&ca_process;#url_In"/>
      <grounding:wsdlMessagePart>
        <xsd:anyURI rdf:value="&ClassAKTGroundingWSDL;#url"/>
      </grounding:wsdlMessagePart>
    </grounding:Wsd1InputMessageMap>
  </grounding:wsdlInputs>

  <!-- Mapping of OWL-S outputs to WSDL message parts -->
  <grounding:wsdlOutputs rdf:parseType="Collection">
    <grounding:Wsd1OutputMessageMap>
      <grounding:daml1Parameter rdf:resource="&ca_process;#classes_Out"/>
      <grounding:wsdlMessagePart>
        <xsd:anyURI rdf:value="&ClassAKTGroundingWSDL;#classes"/>
      </grounding:wsdlMessagePart>
    </grounding:Wsd1OutputMessageMap>
  </grounding:wsdlOutputs>

  <grounding:wsdlReference>
    <xsd:anyURI rdf:value="http://www.w3.org/TR/2001/NOTE-wsdl-
20010315"/>
  </grounding:wsdlReference>
</grounding:Wsd1AtomicProcessGrounding>

<grounding:Wsd1OperationRef rdf:ID="ClassAKT_Operation">
  <rdfs:comment>
    A pointer to the WSDL operation used for ClassAKT
  </rdfs:comment>
  <!-- locate port type to be used -->
  <grounding:portType>
    <xsd:anyURI rdf:value="&ClassAKTGroundingWSDL;#ClassAKT_PortType"/>
  </grounding:portType>
  <!-- locate operation to be used -->
  <grounding:operation>
    <xsd:anyURI rdf:value="&ClassAKTGroundingWSDL;#ClassAKT_Operation"/>

```

```

    </grounding:operation>
  </grounding:WSDLOperationRef>

</rdf:RDF>

```

A.6 Service WSDL Grounding

```

<?xml version="1.0"?>
  <definitions name="ClassAKT_WSDL"
    targetNamespace="http://classakt.actors.org/services/owl-
s/0.9/ClassAKTGrounding.wSDL"
    xmlns:tns="http://classakt.actors.org/services/owl-
s/0.9/ClassAKTGrounding.wSDL"
    xmlns:ClassAKT="http://classakt.actors.org/services/owl-
s/0.9/ClassAKTProcess.owl"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd1="http://classakt.actors.org/services/owl-s/0.9/ClassAKT.xsd"
    xmlns="http://schemas.xmlsoap.org/wsdl/">

    <!-- Types -->

    <types>
      <xsd:schema targetNamespace="http://classakt.actors.org/services/owl-
s/0.9/ClassAKT.xsd">
        <xsd:element name="classes" type="ClassesType"/>
        <xsd:complexType name="ClassesType">
          <xsd:sequence>
            <xsd:element name="class" type="xsd:anyURI"
minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:schema>
    </types>

    <!-- Definition of messages -->

    <!-- Messages -->

    <message name="ClassAKT_Query">
      <part name="url" type="xsd:string" daml-s-parameter="ClassAKT:#url_In"/>
    </message>

    <message name="ClassAKT_Response">
      <part name="classes" element="xsd1:ClassesType" daml-s-
parameter="ClassAKT:#classes_Out"/>
    </message>

    <!-- Ports Definitions -->

    <!-- Port specification -->

    <portType name="ClassAKT_PortType">
      <operation name="ClassAKT_Operation" daml-s-process="ClassAKT:#ClassAKT">
        <input message="tns:ClassAKT_Query"/>
        <output message="tns:ClassAKT_Response"/>
      </operation>
    </portType>

    <!-- Definition of Bindings -->

```

```

<!-- Binding specification -->

<binding name="ClassAKT_httpBinding" type="tns:ClassAKT_PortType">
  <http:binding verb="GET"/>
  <operation name="Classifier">
    <http:operation location="classifier/index.jsp"/>
    <input>
      <http:urlReplacement/>
    </input>
    <output>
      <mime:content type="text/xml"/>
    </output>
  </operation>
</binding>

<!-- Definition of Service -->

<service name="ClassAKT_Service">
  <documentation>
    WSDL description of the ClassAKT service.
  </documentation>

  <!-- the port -->
  <port name="ClassAKT_Port" binding="tns:ClassAKT_httpBinding">
    <http:address location="http://classakt.actors.org/">
  </port>
</service>

</definitions>

```

Appendix B An Ontology for a Digital Archive

B.1 Guidance Notes

Section B.2 lists the ontology that has been defined to describe digital archives. Digital archives together form an important domain that will be an important client of text classifier services on the Semantic Web. Refer to section 7.2 for a discussion of the issues that motivated the design of this ontology.

The ontology was created by the author in a basic text editor. Its validity was confirmed during practical implementation of the Semantic Web text classifier services where, as part of the process, this ontology was consumed by OWL-S API.

B.2 The OWL Ontology

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE daml [
  <!-- namespace URIs -->
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY dc "http://purl.org/dc/elements/1.1/">
  <!ENTITY dct "http://purl.org/dc/terms/">
  <!ENTITY abc "http://metadata.net/harmony/ABC#">
  <!ENTITY dav "http://classakt.aktors.org/ontologies/digital_archive#">
  <!ENTITY base "http://classakt.aktors.org/ontologies/digital_archive#">
]>

<rdf:RDF xmlns="&dav;" xmlns:rdf="&rdf;"
  xmlns:rdfs="&rdfs;" xmlns:owl="&owl;"
  xmlns:xsd="&xsd;" xmlns:abc="&abc;"
  xmlns:dav="&dav;" xmlns:dc="&dc;"
  xmlns:dct="&dct;"
  xml:base="&base;">
```

```

<owl:Ontology rdf:about="">
  <rdfs:label>Digital Archive Reference Ontology (Support
Ontology)</rdfs:label>
  <dc:title xml:lang="en">Digital Archive Reference Ontology (Support
Ontology)</dc:title>
  <dc:description xml:lang="en">Ontology to represent a digital archive on
the Semantic Web. Supports classifications</dc:description>
  <dc:creator>Stephen Ball</dc:creator>
  <dct:created>2005-02-18</dct:created>
  <owl:versionInfo>0.1</owl:versionInfo>
</owl:Ontology>

<owl:Class rdf:ID="IntellectualContent">
  <rdfs:label>IntellectualContent</rdfs:label>
  <rdfs:comment>The class identifying a piece of intellectual
content</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&abc;Work"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="has-title">
  <rdfs:subPropertyOf rdf:resource="&dc;title" />
  <rdfs:label>title of the Intellectual Content</rdfs:label>
  <rdfs:domain rdf:resource="IntellectualContent" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:Class rdf:ID="Classification">
  <rdfs:label>Classification</rdfs:label>
  <rdfs:comment>The classification of a document according to some
scheme</rdfs:comment>
</owl:Class>

<owl:DatatypeProperty rdf:ID="has-ref">
  <rdfs:subPropertyOf rdf:resource="&dc;title" />
  <rdfs:label>A reference value for a given classification</rdfs:label>
  <rdfs:domain rdf:resource="Classification" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="has-title">
  <rdfs:subPropertyOf rdf:resource="&dc;title" />
  <rdfs:label>a title for a given classification</rdfs:label>
  <rdfs:domain rdf:resource="Classification" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:TransitiveProperty rdf:ID="is-sub-classification-of">
  <rdfs:comment>Relationship between classifications in a
hierarchy</rdfs:comment>
  <rdfs:domain rdf:resource="Classification" />
  <rdfs:range rdf:resource="Classification" />
</owl:TransitiveProperty>

<owl:ObjectProperty rdf:ID="has-classification">
  <rdfs:subPropertyOf rdf:resource="&dc;subject" />
  <rdfs:domain rdf:resource="Document" />
  <rdfs:range rdf:resource="Classification" />
</owl:ObjectProperty>

<owl:Class rdf:ID="Document">
  <!-- rdf:about will point at the document -->
  <rdfs:label>Document</rdfs:label>

```

```

    <rdfs:comment>a document accessible via a url on the Web</rdfs:comment>
    <rdfs:subClassOf rdf:resource="&abc;Manifestation"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="has-location">
  <rdfs:subPropertyOf rdf:resource="&dc;Identifier" />
  <rdfs:domain rdf:resource="Document" />
  <!-- points to the physical document on the Web -->
  <rdfs:range rdf:resource="&xsd:anyURI" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="has-format">
  <rdfs:subPropertyOf rdf:resource="&dc;format" />
  <rdfs:domain rdf:resource="Document" />
  <!-- format is expected to be a MIME type -->
  <rdfs:range rdf:resource="&xsd:string" />
</owl:DatatypeProperty>

</rdf:RDF>

```

Appendix C An Ontology for the ACM Portal

C.1 Guidance Notes

Section C.2 lists the ontology that has been defined to extend the ontology for digital archives listed in Appendix B. It specialises the general ontology for digital archives by introducing the classifications defined by the ACM. Refer to section 7.3 for a discussion of the issues that motivated the design of this ontology.

The ontology was created by the author in a basic text editor. Its validity was confirmed during practical implementation of the Semantic Web text classifier services where, as part of the process, this ontology was consumed by OWL-S API.

Space has not permitted a complete listing of this ontology. Instead, simply those classifications required for the ‘A’ section of the ACM classification scheme are shown for illustration.

C.2 The OWL Ontology

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE daml [
  <!-- namespace URIs -->
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY dc "http://purl.org/dc/elements/1.1/">
  <!ENTITY dct "http://purl.org/dc/terms/">
  <!ENTITY dav "http://classakt.aktors.org/ontologies/digital_archive#">
  <!ENTITY davacm "http://classakt.aktors.org/ontologies/digital_archive_acm#">
  <!ENTITY acm "http://www.acm.org/class/1998/">
]>
```

```

<rdf:RDF xmlns="&davacm;" xmlns:rdf="&rdf;"
  xmlns:rdfs="&rdfs;" xmlns:owl="&owl;"
  xmlns:xsd="&xsd;" xmlns:dav="&dav;"
  xmlns:davacm="&davacm;" xmlns:acm="&acm;"
  xmlns:dc="&dc;" xmlns:dct="&dct;">

<owl:Ontology rdf:about="">
  <rdfs:label>Digital Archive Ontology with ACM classification</rdfs:label>
  <dc:title xml:lang="en">Digital Archive Ontology with ACM
Classification</dc:title>
  <dc:description xml:lang="en">Ontology representing a digital archives
supporting ACM classification</dc:description>
  <dc:creator>Stephen Ball</dc:creator>
  <dct:created>2005-02-18</dct:created>
  <owl:versionInfo>0.1.1</owl:versionInfo>
</owl:Ontology>

<owl:Class rdf:about="&acm;ACM-Classification">
  <rdfs:label>ACM Classification</rdfs:label>
  <rdfs:comment>Class representing various classifications described by the
ACM</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&dav;Classification"/>
</owl:Class>
<acm:ACM-Classification rdf:about="&acm;A">
  <dav:has-ref>A</dav:has-ref>
  <dav:has-title>General Literature</dav:has-title>
</acm:ACM-Classification>
<acm:ACM-Classification rdf:about="&acm;A.0">
  <dav:has-ref>A.0</dav:has-ref>
  <dav:has-title>GENERAL</dav:has-title>
  <dav:is-sub-classification-of rdf:resource="&acm;A"/>
</acm:ACM-Classification>
<acm:ACM-Classification rdf:about="&acm;A.0-biographies-autobiographies">
  <dav:has-ref>A.0</dav:has-ref>
  <dav:has-title>Biographies/autobiographies</dav:has-title>
  <dav:is-sub-classification-of rdf:resource="&acm;A.0"/>
</acm:ACM-Classification>
<acm:ACM-Classification rdf:about="&acm;A.0-conference-proceedings">
  <dav:has-ref>A.0</dav:has-ref>
  <dav:has-title>Conference proceedings</dav:has-title>
  <dav:is-sub-classification-of rdf:resource="&acm;A.0"/>
</acm:ACM-Classification>
<acm:ACM-Classification rdf:about="&acm;A.0-general-literary-works">
  <dav:has-ref>A.0</dav:has-ref>
  <dav:has-title>General literary works</dav:has-title>
  <dav:is-sub-classification-of rdf:resource="&acm;A.0"/>
  <rdfs:comment>e.g., fiction, plays</rdfs:comment>
</acm:ACM-Classification>
<acm:ACM-Classification rdf:about="&acm;A.1">
  <dav:has-ref>A.1</dav:has-ref>
  <dav:has-title>INTRODUCTORY AND SURVEY</dav:has-title>
  <dav:is-sub-classification-of rdf:resource="&acm;A"/>
</acm:ACM-Classification>
<acm:ACM-Classification rdf:about="&acm;A.2">
  <dav:has-ref>A.2</dav:has-ref>
  <dav:has-title>REFERENCE</dav:has-title>
  <dav:is-sub-classification-of rdf:resource="&acm;A"/>
  <rdfs:comment>e.g., dictionaries, encyclopedias, glossaries</rdfs:comment>
</acm:ACM-Classification>
<acm:ACM-Classification rdf:about="&acm;A.m">
  <dav:has-ref>A.m</dav:has-ref>
  <dav:has-title>MISCELLANEOUS</dav:has-title>
  <dav:is-sub-classification-of rdf:resource="&acm;A"/>

```

```
</acm:ACM-Classification>  
    ...etc for other ACM classes  
</rdf:RDF>
```

Appendix D An Ontology for Text Classification

D.1 Guidance Notes

Section D.2 lists the ontology that has been defined to enable a service orientated architecture for feature generators and text classifiers. Refer to section 7.4 for a discussion of the issues that motivated the design of this ontology.

The ontology was created by the author in a basic text editor. Its validity was confirmed during practical implementation of the Semantic Web text classifier services where, as part of the process, this ontology was consumed by OWL-S API.

D.2 The OWL Ontology

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE daml [
  <!-- namespace URIs -->
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY dc "http://purl.org/dc/elements/1.1/">
  <!ENTITY dct "http://purl.org/dc/terms/">
  <!ENTITY tc "http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#">
]>

<rdf:RDF xmlns:rdf="&rdf;"
  xmlns:rdfs="&rdfs;" xmlns:owl="&owl;"
  xmlns:xsd="&xsd;"
  xmlns:tc="&tc;" xmlns:dc="&dc;"
  xmlns:dct="&dct;">

  <owl:Ontology rdf:about="">
    <rdfs:label>Ontology for Text Classification</rdfs:label>
    <dc:title xml:lang="en">Ontology for Text Classification</dc:title>
```

```

    <dc:description xml:lang="en">Ontology to describe text classifiers and
feature generators on the Semantic Web</dc:description>
    <dc:creator>Stephen Ball</dc:creator>
    <dc:created>2005-06-21</dc:created>
    <owl:versionInfo>0.1</owl:versionInfo>
</owl:Ontology>

<owl:Class rdf:ID="Document">
    <rdfs:label>Document</rdfs:label>
    <rdfs:comment>A document</rdfs:comment>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
            <owl:onProperty>
                <owl:FunctionalProperty rdf:about="#hasDataFormat"/>
            </owl:onProperty>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
            <owl:onProperty>
                <owl:FunctionalProperty rdf:about="#hasMIME-Type"/>
            </owl:onProperty>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="has-location">
    <rdfs:domain rdf:resource="Document" />
    <!-- points to the physical document on the Web -->
    <rdfs:range rdf:resource="&xsd:anyURI" />
</owl:DatatypeProperty>

<owl:Class rdf:ID="WebDocument">
    <rdfs:label>Web Document</rdfs:label>
    <rdfs:comment>a document that exists on the Internet</rdfs:comment>
    <rdfs:subClassOf>
        <owl:Class rdf:about="#Document"/>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="FeaturesDocument">
    <rdfs:label>Features Document</rdfs:label>
    <rdfs:comment>A document containing a set of features</rdfs:comment>
    <rdfs:subClassOf>
        <owl:Class rdf:about="#Document"/>
    </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="has-number-of-features">
    <rdfs:label>Number of features</rdfs:label>
    <rdfs:domain rdf:resource="FeaturesDocument" />
    <rdfs:range rdf:resource="&xsd:int" />
</owl:DatatypeProperty>

<owl:Class rdf:ID="MIME-Type">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <Annotation rdf:about="#text-plain"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>

```

```

        <Annotation rdf:about="#text-xml"/>
        <Annotation rdf:about="#application-pdf"/>
    </owl:oneOf>
</owl:Class>
</owl:equivalentClass>
</owl:Class>

<owl:Class rdf:ID="DataFormat">
    <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Describes the concrete
syntax in which data and annotation are encoded. In particular, specifies how
the units of various types are demarcated.</rdfs:comment>
</owl:Class>

<owl:Class rdf:ID="Annotation">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <Annotation rdf:about="#Raw"/>
                <Annotation rdf:about="#Stopped"/>
                <Annotation rdf:about="#Stemmed"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:ID="Language">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <Language rdf:about="#en"/>
                <Language rdf:ID="fr"/>
                <Language rdf:ID="de"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>

<owl:FunctionalProperty rdf:ID="hasMIME-Type">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="#MIME-Type"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>

<owl:FunctionalProperty rdf:ID="hasDataFormat">
    <rdfs:domain rdf:resource="#Document"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#DataFormat"/>
</owl:FunctionalProperty>

<owl:ObjectProperty rdf:ID="hasAnnotation">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="#Annotation"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasSubjectLanguage">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">The
language which the content of the resource describes or
discusses.</rdfs:comment>
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="#Language"/>
</owl:ObjectProperty>

```

```

<owl:Class rdf:ID="Classification">
  <rdfs:label>Classification</rdfs:label>
  <rdfs:comment>The classification of a document according to some
scheme</rdfs:comment>
</owl:Class>

<owl:DatatypeProperty rdf:ID="has-ref">
  <rdfs:subPropertyOf rdf:resource="&dc:title" />
  <rdfs:label>A reference value for a given classification</rdfs:label>
  <rdfs:domain rdf:resource="Classification" />
  <rdfs:range rdf:resource="&xsd:string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="has-title">
  <rdfs:subPropertyOf rdf:resource="&dc:title" />
  <rdfs:label>a title for a given classification</rdfs:label>
  <rdfs:domain rdf:resource="Classification" />
  <rdfs:range rdf:resource="&xsd:string" />
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="has-classification">
  <rdfs:domain rdf:resource="Document" />
  <rdfs:range rdf:resource="Classification" />
</owl:ObjectProperty>

<owl:Class rdf:ID="TC-Resource">
  <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-
s/1.0/Profile.owl#Profile"/>
</owl:Class>

<owl:Class rdf:ID="TC-ProcessingResource">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#WebDocument"/>
      <owl:onProperty rdf:resource="http://www.daml.org/services/owl-
s/1.0/Profile.owl#hasInput"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#Classification"/>
      <owl:onProperty rdf:resource="http://www.daml.org/services/owl-
s/1.0/Profile.owl#hasInput"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#NL-Resource"/>
</owl:Class>

<owl:Class rdf:ID="TC-FeatureExtractor">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#FeaturesDocument"/>
      <owl:onProperty rdf:resource="http://www.daml.org/services/owl-
s/1.0/Profile.owl#hasOutput"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#TC-ProcessingResource"/>
</owl:Class>

<owl:Class rdf:ID="TC-ClassifierGenerator">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#FeaturesDocument"/>

```

```
        <owl:onProperty rdf:resource="http://www.daml.org/services/owl-
s/1.0/Profile.owl#hasInput"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#TC-ProcessingResource"/>
</owl:Class>

</rdf:RDF>
```

Appendix E An Ontology for Classifier Service Generation

E.1 Guidance Notes

This section of the appendix lists the various ontologies for atomic services required to effect service generation of text classifier services. It should be considered in conjunction with Figure 8.3 which describes the composite service that orchestrates these various atomic services. In addition to the core atomic services listed in Appendices E.2 to E.6, E.7 is a ‘template’ ontology used by the Web service generator to create an ontology for the atomic service that it generates.

These ontologies were created by the author in a basic text editor using, as a starting point, example service ontologies made available by the authors of OWL-S API. Functioning Semantic Web services were introduced in section 8.3 and the listings below form the OWL-S interface to the *ClassifiersServiceEx* Web service. They have been demonstrated to be correct and fit for purpose by a client program written in Java that consumes the service interfaces using OWL-S API and invokes the services to generate new text classifier Semantic Web service as required.

The atomic services that make up the feature generation process, *FeatureCreationService* as indicated in Figure 8.2, have not been listed in this thesis. In practice, they are very similar to the services listed here for the classification process.

E.2 ClassifierCreation_CreateSession

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
  <!ENTITY tc-profile "http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
]>

<rdf:RDF
  xmlns:owl=          "&owl;#"
  xmlns:rdfs=         "&rdfs;#"
  xmlns:rdf=          "&rdf;#"
  xmlns:service=      "&service;#"
  xmlns:process=      "&process;#"
  xmlns:tc-profile=   "&tc-profile;#"
  xmlns:grounding=    "&grounding;#"
  xmlns:xsd=          "&xsd;#"
  xml:base=           "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/ClassifierCreation_CreateSession.owl"
  xmlns=              "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/ClassifierCreation_CreateSession.owl#"
>

<!-- Service description -->
<service:Service rdf:ID="ClassifierCreation_CreateSession_Service">
  <service:presents
rdf:resource="#ClassifierCreation_CreateSession_Profile"/>

    <service:describedBy
rdf:resource="#ClassifierCreation_CreateSession_ProcessModel"/>

    <service:supports
rdf:resource="#ClassifierCreation_CreateSession_Grounding"/>
</service:Service>

<!-- Profile description -->
<tc-profile:TC-Resource rdf:ID="ClassifierCreation_CreateSession_Profile">
  <service:isPresentedBy
rdf:resource="#ClassifierCreation_CreateSession_Service"/>

    <tc-profile:serviceName xml:lang="en">Create Session for classifier
creation</tc-profile:serviceName>
    <tc-profile:textDescription xml:lang="en">Returns a session id</tc-
profile:textDescription>

    <tc-profile:hasOutput rdf:resource="#Session-id"/>
</tc-profile:TC-Resource>

<!-- Process Model description -->
<process:AtomicProcess rdf:ID="ClassifierCreation_CreateSession_ProcessModel">
  <service:describes
rdf:resource="#ClassifierCreation_CreateSession_Service"/>

    <process:hasOutput rdf:resource="#Session-id"/>
</process:AtomicProcess>
```

```

<process:Output rdf:ID="Session-id">
  <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/process_
management#session-id</process:parameterType>
    <rdfs:label>Session id</rdfs:label>
</process:Output>

<!-- Grounding description -->
<grounding:Wsd1Grounding rdf:ID="ClassifierCreation_CreateSession_Grounding">
  <service:supportedBy
rdf:resource="#ClassifierCreation_CreateSession_Service"/>
  <grounding:hasAtomicProcessGrounding
rdf:resource="#ClassifierCreation_CreateSession_ProcessGrounding"/>
</grounding:Wsd1Grounding>

<grounding:Wsd1AtomicProcessGrounding
rdf:ID="ClassifierCreation_CreateSession_ProcessGrounding">
  <grounding:owlsProcess
rdf:resource="#ClassifierCreation_CreateSession_ProcessModel"/>
  <grounding:wsdlDocument>http://classakt.aktors.org:8000/ClassifiersServic
eEx/ClassifiersServiceEx.dll?Handler=GenClassifiersServiceExWSDL</grounding:wsd
lDocument>
  <grounding:wsdlOperation>
    <grounding:Wsd1OperationRef>

    <grounding:portType>urn:ClassifiersServiceExService#ClassifiersServiceExS
erviceSoap</grounding:portType>

    <grounding:operation>urn:ClassifiersServiceExService#CreateSession</groun
ding:operation>
      </grounding:Wsd1OperationRef>
    </grounding:wsdlOperation>

    <grounding:wsdlOutputMessage
rdf:datatype="&xsd:anyURI">urn:ClassifiersServiceExService#CreateSessionOut</g
rounding:wsdlOutputMessage>
      <grounding:wsdlOutput>
        <grounding:Wsd1OutputMessageMap>
          <grounding:owlsParameter rdf:resource="#Session-id"/>

          <grounding:wsdlMessagePart>urn:ClassifiersServiceExService#return</ground
ing:wsdlMessagePart>
            <grounding:xsltTransformationString>
              <![CDATA[
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/">
    <xsl:variable name="X1" select="*" />
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:pm-
ont="http://www.classakt.aktors.org:8000/ontologies/process_management#">
      <pm-ont:session-id>
        <pm-ont:has-session-id><xsl:value-of
select="$X1"/></pm-ont:has-session-id>
      </pm-ont:session-id>
    </rdf:RDF>
  </xsl:template>
</xsl:stylesheet>
      ]]>
            </grounding:xsltTransformationString>
          </grounding:Wsd1OutputMessageMap>
        </grounding:wsdlOutput>
      </grounding:wsdlOutput>

```

</grounding:WsdlAtomicProcessGrounding>

</rdf:RDF>

E.3 ClassifierCreation_AddClass

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
  <!ENTITY tc-profile "http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
  <!ENTITY pm
"http://classakt.aktors.org:8000/ontologies/process_management.owl">
  <!ENTITY tc "http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl">
]>

<rdf:RDF
  xmlns:owl=          "&owl;#"
  xmlns:rdfs=         "&rdfs;#"
  xmlns:rdf=          "&rdf;#"
  xmlns:service=      "&service;#"
  xmlns:process=      "&process;#"
  xmlns:tc-profile=   "&tc-profile;#"
  xmlns:grounding=    "&grounding;#"
  xmlns:pm=           "&pm;#"
  xmlns:tc=           "&tc;#"
  xmlns:xsd=          "&xsd;#"
  xml:base=           "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/ClassifierCreation_AddClass.owl"
  xmlns=              "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/ClassifierCreation_AddClass.owl#"
>

<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="&service;"/>
  <owl:imports rdf:resource="&tc-profile;"/>
  <owl:imports rdf:resource="&process;"/>
  <owl:imports rdf:resource="&grounding;"/>
  <owl:imports rdf:resource="&pm;"/>
  <owl:imports rdf:resource="&tc;"/>
</owl:Ontology>

<!-- Service description -->
<service:Service rdf:ID="ClassifierCreation_AddClass_Service">
  <service:presents rdf:resource="#ClassifierCreation_AddClass_Profile"/>

  <service:describedBy
rdf:resource="#ClassifierCreation_AddClass_ProcessModel"/>

  <service:supports rdf:resource="#ClassifierCreation_AddClass_Grounding"/>
</service:Service>

<!-- Profile description -->
<tc-profile:TC-Resource rdf:ID="ClassifierCreation_AddClass_Profile">
```

```

    <service:isPresentedBy
rdf:resource="#ClassifierCreation_AddClass_Service"/>

    <tc-profile:serviceName xml:lang="en">Add Class</tc-profile:serviceName>
    <tc-profile:textDescription xml:lang="en">Add a class required for
Classifier creation</tc-profile:textDescription>

    <tc-profile:hasInput rdf:resource="#Session-id"/>
    <tc-profile:hasInput rdf:resource="#Classification"/>
</tc-profile:TC-Resource>

<!-- Process Model description -->
<process:AtomicProcess rdf:ID="ClassifierCreation_AddClass_ProcessModel">
    <service:describes rdf:resource="#ClassifierCreation_AddClass_Service"/>

    <process:hasInput rdf:resource="#Session-id"/>
    <process:hasInput rdf:resource="#Classification"/>
</process:AtomicProcess>

<process:Input rdf:ID="Session-id">
    <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/process_
management.owl#session-id</process:parameterType>
    <rdfs:label>Session id</rdfs:label>
</process:Input>

<process:Input rdf:ID="Classification">
    <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#Classification</process:parameterType>
    <rdfs:label>Classification</rdfs:label>
</process:Input>

<!-- Grounding description -->
<grounding:WsdgGrounding rdf:ID="ClassifierCreation_AddClass_Grounding">
    <service:supportedBy
rdf:resource="#ClassifierCreation_AddClass_Service"/>
    <grounding:hasAtomicProcessGrounding
rdf:resource="#ClassifierCreation_AddClass_ProcessGrounding"/>
</grounding:WsdgGrounding>

<grounding:WsdgAtomicProcessGrounding
rdf:ID="ClassifierCreation_AddClass_ProcessGrounding">
    <grounding:owlsProcess
rdf:resource="#ClassifierCreation_AddClass_ProcessModel"/>
    <grounding:wsdlDocument>http://classakt.aktors.org:8000/ClassifiersServic
eEx/ClassifiersServiceEx.dll?Handler=GenClassifiersServiceExWSDL</grounding:wsd
lDocument>
    <grounding:wsdlOperation>
        <grounding:WsdgOperationRef>

        <grounding:portType>urn:ClassifiersServiceExService#ClassifiersServiceExS
erviceSoap</grounding:portType>

        <grounding:operation>urn:ClassifiersServiceExService#AddClass</grounding:
operation>
            </grounding:WsdgOperationRef>
        </grounding:wsdlOperation>

        <grounding:wsdlInputMessage
rdf:datatype="&xsd:anyURI">urn:ClassifiersServiceExService#AddClassIn</groundi
ng:wsdlInputMessage>
        <grounding:wsdlInput>

```

```

        <grounding:WsdInputMessageMap>
            <grounding:owlsParameter rdf:resource="#Session-id"/>

        <grounding:wsdlMessagePart>urn:ClassifiersServiceExService#nSessionID</gr
ounding:wsdlMessagePart>
            <grounding:xsltTransformationString>
                <![CDATA[
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:pm="http://classakt.aktors.org:8000/ontologies/process_management.owl#"
  rdf:keepNamespace=""
  pm:keepNamespace="">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/ ">
      <xsl:value-of select="rdf:RDF/pm:session-id/pm:has-session-id"/>
    </xsl:template>
  </xsl:stylesheet>
]]>
            </grounding:xsltTransformationString>
        </grounding:WsdInputMessageMap>
    </grounding:wsdlInput>
    <grounding:wsdlInput>
        <grounding:WsdInputMessageMap>
            <grounding:owlsParameter rdf:resource="#Classification"/>

        <grounding:wsdlMessagePart>urn:ClassifiersServiceExService#strClassID1</g
rounding:wsdlMessagePart>
            <grounding:xsltTransformationString>
                <![CDATA[
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:tc="http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#"
  rdf:keepNamespace=""
  tc:keepNamespace="">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/ ">
      <xsl:value-of select="rdf:RDF/tc:Classification/tc:has-ref"/>
    </xsl:template>
  </xsl:stylesheet>
]]>
            </grounding:xsltTransformationString>
        </grounding:WsdInputMessageMap>
    </grounding:wsdlInput>
    <grounding:wsdlInput>
        <grounding:WsdInputMessageMap>
            <grounding:owlsParameter rdf:resource="#Classification"/>

        <grounding:wsdlMessagePart>urn:ClassifiersServiceExService#strClassID2</g
rounding:wsdlMessagePart>
            <grounding:xsltTransformationString>
                <![CDATA[
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:tc="http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#"
  rdf:keepNamespace=""
  tc:keepNamespace="">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

```

```

        <xsl:template match="/ ">
            <xsl:value-of select="rdf:RDF/tc:Classification/tc:has-title"/>
        </xsl:template>
    </xsl:stylesheet>

    ]]>
    </grounding:xsltTransformationString>
    </grounding:WsdInputMessageMap>
    </grounding:wsdlInput>

</grounding:WsdAtomicProcessGrounding>

</rdf:RDF>

```

E.4 ClassifierCreation_AddFeaturesDocument

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
    <!ENTITY owl "http://www.w3.org/2002/07/owl">
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
    <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
    <!ENTITY tc-profile "http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl">
    <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
    <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
    <!ENTITY pm
"http://classakt.aktors.org:8000/ontologies/process_management.owl">
    <!ENTITY tc "http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl">
]>

<rdf:RDF
    xmlns:owl=          "&owl;#"
    xmlns:rdfs=         "&rdfs;#"
    xmlns:rdf=          "&rdf;#"
    xmlns:service=      "&service;#"
    xmlns:process=      "&process;#"
    xmlns:tc-profile=   "&tc-profile;#"
    xmlns:grounding=    "&grounding;#"
    xmlns:pm=           "&pm;#"
    xmlns:tc=           "&tc;#"
    xmlns:xsd=          "&xsd;#"
    xml:base=           "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/ClassifierCreation_AddFeaturesDocument.owl"
    xmlns=              "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/ClassifierCreation_AddFeaturesDocument.owl#"
>

<owl:Ontology rdf:about="">
    <owl:imports rdf:resource="&service;"/>
    <owl:imports rdf:resource="&tc-profile;"/>
    <owl:imports rdf:resource="&process;"/>
    <owl:imports rdf:resource="&grounding;"/>
    <owl:imports rdf:resource="&pm;"/>
    <owl:imports rdf:resource="&tc;"/>
</owl:Ontology>

<!-- Service description -->
<service:Service rdf:ID="ClassifierCreation_AddFeaturesDocument_Service">
    <service:presents
rdf:resource="#ClassifierCreation_AddFeaturesDocument_Profile"/>

```

```

    <service:describedBy
rdf:resource="#ClassifierCreation_AddFeaturesDocument_ProcessModel"/>

    <service:supports
rdf:resource="#ClassifierCreation_AddFeaturesDocument_Grounding"/>
</service:Service>

<!-- Profile description -->
<tc-profile:TC-Resource
rdf:ID="ClassifierCreation_AddFeaturesDocument_Profile">
    <service:isPresentedBy
rdf:resource="#ClassifierCreation_AddFeaturesDocument_Service"/>

    <tc-profile:serviceName xml:lang="en">Add Features in a
FeaturesDocument</tc-profile:serviceName>
    <tc-profile:textDescription xml:lang="en">Add a feature required for
Classifier creation</tc-profile:textDescription>

    <tc-profile:hasInput rdf:resource="#Session-id"/>
    <tc-profile:hasInput rdf:resource="#Features"/>
</tc-profile:TC-Resource>

<!-- Process Model description -->
<process:AtomicProcess
rdf:ID="ClassifierCreation_AddFeaturesDocument_ProcessModel">
    <service:describes
rdf:resource="#ClassifierCreation_AddFeaturesDocument_Service"/>

    <process:hasInput rdf:resource="#Session-id"/>
    <process:hasInput rdf:resource="#Features"/>
</process:AtomicProcess>

<process:Input rdf:ID="Session-id">
    <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/process_
management.owl#session-id</process:parameterType>
    <rdfs:label>Session id</rdfs:label>
</process:Input>

<process:Input rdf:ID="Features">
    <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#FeaturesDocument</process:parameterType>
    <rdfs:label>Features</rdfs:label>
</process:Input>

<!-- Grounding description -->
<grounding:WsdgGrounding
rdf:ID="ClassifierCreation_AddFeaturesDocument_Grounding">
    <service:supportedBy
rdf:resource="#ClassifierCreation_AddFeaturesDocument_Service"/>
    <grounding:hasAtomicProcessGrounding
rdf:resource="#ClassifierCreation_AddFeaturesDocument_ProcessGrounding"/>
</grounding:WsdgGrounding>

<grounding:WsdgAtomicProcessGrounding
rdf:ID="ClassifierCreation_AddFeaturesDocument_ProcessGrounding">
    <grounding:owlsProcess
rdf:resource="#ClassifierCreation_AddFeaturesDocument_ProcessModel"/>
    <grounding:wsdlDocument>http://classakt.aktors.org:8000/ClassifiersServic
eEx/ClassifiersServiceEx.dll?Handler=GenClassifiersServiceExWSDL</grounding:wsd
lDocument>

```

```

    <grounding:wSDLOperation>
      <grounding:WSDLOperationRef>

    <grounding:portType>urn:ClassifiersServiceExService#ClassifiersServiceExS
erviceSoap</grounding:portType>

    <grounding:operation>urn:ClassifiersServiceExService#AddFeaturesDocument<
/grounding:operation>
      </grounding:WSDLOperationRef>
    </grounding:wSDLOperation>

    <grounding:wSDLInputMessage
rdf:datatype="&xsd:anyURI">urn:ClassifiersServiceExService#AddFeatureIn</groun
ding:wSDLInputMessage>
      <grounding:wSDLInput>
        <grounding:WSDLInputMessageMap>
          <grounding:owlsParameter rdf:resource="#Session-id"/>

        <grounding:wSDLMessagePart>urn:ClassifiersServiceExService#nSessionID</gr
ounding:wSDLMessagePart>
          <grounding:xsltTransformationString>
            <![CDATA[
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:pm="http://classakt.aktors.org:8000/ontologies/process_management.owl#"
  rdf:keepNamespace=""
  pm:keepNamespace="">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/" >
      <xsl:value-of select="rdf:RDF/pm:session-id/pm:has-session-id"/>
    </xsl:template>
  </xsl:stylesheet>
    ]]>
          </grounding:xsltTransformationString>
        </grounding:WSDLInputMessageMap>
      </grounding:wSDLInput>
    </grounding:wSDLInput>
      <grounding:WSDLInputMessageMap>
        <grounding:owlsParameter rdf:resource="#Features"/>

    <grounding:wSDLMessagePart>urn:ClassifiersServiceExService#strFeaturesDoc
umentURL</grounding:wSDLMessagePart>
      <grounding:xsltTransformationString>
        <![CDATA[
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:tc="http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#"
  rdf:keepNamespace=""
  tc:keepNamespace="">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/" >
      <xsl:value-of select="rdf:RDF/tc:FeaturesDocument/tc:has-
location"/>
    </xsl:template>
  </xsl:stylesheet>
    ]]>
      </grounding:xsltTransformationString>
    </grounding:WSDLInputMessageMap>
  </grounding:wSDLInput>

```

</grounding:WsdAtomicProcessGrounding>

</rdf:RDF>

E.5 ClassifierCreation_AddInstance

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
  <!ENTITY tc-profile "http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
  <!ENTITY pm
"http://classakt.aktors.org:8000/ontologies/process_management.owl">
  <!ENTITY tc "http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl">
]>

<rdf:RDF
  xmlns:owl=          "&owl;#"
  xmlns:rdfs=         "&rdfs;#"
  xmlns:rdf=          "&rdf;#"
  xmlns:service=      "&service;#"
  xmlns:process=      "&process;#"
  xmlns:tc-profile=   "&tc-profile;#"
  xmlns:grounding=    "&grounding;#"
  xmlns:pm=           "&pm;#"
  xmlns:tc=           "&tc;"
  xmlns:xsd=          "&xsd;#"
  xml:base=           "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/FeatureCreation_AddInstance.owl"
  xmlns=              "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/FeatureCreation_AddInstance.owl#"
>

  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="&service;"/>
    <owl:imports rdf:resource="&tc-profile;"/>
    <owl:imports rdf:resource="&process;"/>
    <owl:imports rdf:resource="&grounding;"/>
    <owl:imports rdf:resource="&pm;"/>
    <owl:imports rdf:resource="&tc;"/>
  </owl:Ontology>

  <!-- Service description -->
  <service:Service rdf:ID="ClassifierCreation_AddInstance_Service">
    <service:presents
rdf:resource="#ClassifierCreation_AddInstance_Profile"/>

    <service:describedBy
rdf:resource="#ClassifierCreation_AddInstance_ProcessModel"/>

    <service:supports
rdf:resource="#ClassifierCreation_AddInstance_Grounding"/>
  </service:Service>
```

```

<!-- Profile description -->
<tc-profile:TC-Resource rdf:ID="ClassifierCreation_AddInstance_Profile">
  <service:isPresentedBy
rdf:resource="#ClassifierCreation_AddInstance_Service"/>

  <tc-profile:serviceName xml:lang="en">Add Instance</tc-
profile:serviceName>
  <tc-profile:textDescription xml:lang="en">Add text for a given class</tc-
profile:textDescription>

  <tc-profile:hasInput rdf:resource="#Session-id"/>
  <tc-profile:hasInput rdf:resource="#Document"/>
  <tc-profile:hasInput rdf:resource="#Classification"/>
</tc-profile:TC-Resource>

<!-- Process Model description -->
<process:AtomicProcess rdf:ID="ClassifierCreation_AddInstance_ProcessModel">
  <service:describes
rdf:resource="#ClassifierCreation_AddInstance_Service"/>

  <process:hasInput rdf:resource="#Session-id"/>
  <process:hasInput rdf:resource="#Document"/>
  <process:hasInput rdf:resource="#Classification"/>
</process:AtomicProcess>

<process:Input rdf:ID="Session-id">
  <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/process_
management.owl#session-id</process:parameterType>
  <rdfs:label>Session id</rdfs:label>
</process:Input>

<process:Input rdf:ID="Document">
  <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#Document</process:parameterType>
  <rdfs:label>Instance text</rdfs:label>
</process:Input>

<process:Input rdf:ID="Classification">
  <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy#Classification</process:parameterType>
  <rdfs:label>Class id number 1</rdfs:label>
</process:Input>

<!-- Grounding description -->
<grounding:WsdgGrounding rdf:ID="ClassifierCreation_AddInstance_Grounding">
  <service:supportedBy
rdf:resource="#ClassifierCreation_AddInstance_Service"/>
  <grounding:hasAtomicProcessGrounding
rdf:resource="#ClassifierCreation_AddInstance_ProcessGrounding"/>
</grounding:WsdgGrounding>

<grounding:WsdgAtomicProcessGrounding
rdf:ID="ClassifierCreation_AddInstance_ProcessGrounding">
  <grounding:owlsProcess
rdf:resource="#ClassifierCreation_AddInstance_ProcessModel"/>
  <grounding:wsdlDocument>http://classakt.aktors.org:8000/ClassifiersServic
eEx/ClassifiersServiceEx.dll?Handler=GenClassifiersServiceExWSDL</grounding:wsd
lDocument>
  <grounding:wsdlOperation>
    <grounding:WsdgOperationRef>

```

```

    <grounding:portType>urn:ClassifiersServiceExService#ClassifiersServiceExServiceSoap</grounding:portType>

    <grounding:operation>urn:ClassifiersServiceExService#AddInstance</grounding:operation>
      </grounding:WsdloperationRef>
    </grounding:wsdlOperation>

    <grounding:wsdlInputMessage
rdf:datatype="&xsd:anyURI">urn:ClassifiersServiceExService#AddInstanceIn</grounding:wsdlInputMessage>
      <grounding:wsdlInput>
        <grounding:WsdlinputMessageMap>
          <grounding:owlsParameter rdf:resource="#Session-id"/>

          <grounding:wsdlMessagePart>urn:ClassifiersServiceExService#nSessionID</grounding:wsdlMessagePart>
            <grounding:xsltTransformationString>
              <![CDATA[
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:pm="http://classakt.aktors.org:8000/ontologies/process_management.owl#"
  rdf:keepNamespace=""
  pm:keepNamespace="">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/" >
      <xsl:value-of select="rdf:RDF/pm:session-id/pm:has-session-id"/>
    </xsl:template>
  </xsl:stylesheet>
    ]]>
            </grounding:xsltTransformationString>
          </grounding:WsdlinputMessageMap>
        </grounding:wsdlInput>
      <grounding:wsdlInput>
        <grounding:WsdlinputMessageMap>
          <grounding:owlsParameter rdf:resource="#Document"/>

          <grounding:wsdlMessagePart>urn:ClassifiersServiceExService#strInstanceText</grounding:wsdlMessagePart>
            <grounding:xsltTransformationString>
              <![CDATA[
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:tc="http://classakt.aktors.org:8000/ontologies/TC-ProfileHierarchy.owl#"
  rdf:keepNamespace=""
  tc:keepNamespace="">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/" >
      <xsl:value-of select="rdf:RDF/tc:Document/tc:has-location"/>
    </xsl:template>
  </xsl:stylesheet>
    ]]>
            </grounding:xsltTransformationString>
          </grounding:WsdlinputMessageMap>
        </grounding:wsdlInput>
      <grounding:wsdlInput>
        <grounding:WsdlinputMessageMap>
          <grounding:owlsParameter rdf:resource="#Classification"/>

```

```

        <grounding:wsdlMessagePart>urn:ClassifiersServiceExService#strClassID1</g
rounding:wsdlMessagePart>
        <grounding:xsltTransformationString>
        <![CDATA[
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:tc="http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#"
  rdf:keepNamespace=""
  tc:keepNamespace="">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/ ">
    <xsl:value-of select="rdf:RDF/tc:Classification/tc:has-ref"/>
  </xsl:template>
</xsl:stylesheet>
        ]]>
        </grounding:xsltTransformationString>
        </grounding:WsdInputMessageMap>
        </grounding:wsdlInput>
        <grounding:wsdlInput>
        <grounding:WsdInputMessageMap>
        <grounding:owlsParameter rdf:resource="#Classification"/>

        <grounding:wsdlMessagePart>urn:ClassifiersServiceExService#strClassID2</g
rounding:wsdlMessagePart>
        <grounding:xsltTransformationString>
        <![CDATA[
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:tc="http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#"
  rdf:keepNamespace=""
  tc:keepNamespace="">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/ ">
    <xsl:value-of select="rdf:RDF/tc:Classification/tc:has-title"/>
  </xsl:template>
</xsl:stylesheet>
        ]]>
        </grounding:xsltTransformationString>
        </grounding:WsdInputMessageMap>
        </grounding:wsdlInput>

</grounding:WsdAtomicProcessGrounding>

</rdf:RDF>

```

E.6 ClassifierCreation_CreateClassifier

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
  <!ENTITY tc-profile "http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">

```

```

    <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
    <!ENTITY pm
"http://classakt.aktors.org:8000/ontologies/process_management.owl">
]>

```

```

<rdf:RDF
  xmlns:owl=          "&owl;#"
  xmlns:rdfs=         "&rdfs;#"
  xmlns:rdf=          "&rdf;#"
  xmlns:service=      "&service;#"
  xmlns:process=      "&process;#"
  xmlns:tc-profile=   "&tc-profile;#"
  xmlns:grounding=    "&grounding;#"
  xmlns:pm=           "&pm;#"
  xmlns:xsd=          "&xsd;#"
  xml:base=           "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/ClassifierCreation_CreateClassifier.owl"
  xmlns=              "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/ClassifierCreation_CreateClassifier.owl#"
>

```

```

<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="&service;"/>
  <owl:imports rdf:resource="&tc-profile;"/>
  <owl:imports rdf:resource="&process;"/>
  <owl:imports rdf:resource="&grounding;"/>
  <owl:imports rdf:resource="&pm;"/>
</owl:Ontology>

```

```

<!-- Service description -->
<service:Service rdf:ID="ClassifierCreation_CreateClassifier_Service">
  <service:presents
rdf:resource="#ClassifierCreation_CreateClassifier_Profile"/>

  <service:describedBy
rdf:resource="#ClassifierCreation_CreateClassifier_ProcessModel"/>

  <service:supports
rdf:resource="#ClassifierCreation_CreateClassifier_Grounding"/>
</service:Service>

```

```

<!-- Profile description -->
<tc-profile:TC-Resource rdf:ID="ClassifierCreation_CreateClassifier_Profile">
  <service:isPresentedBy
rdf:resource="#ClassifierCreation_CreateClassifier_Service"/>

  <tc-profile:serviceName xml:lang="en">CreateClassifier</tc-
profile:serviceName>
  <tc-profile:textDescription xml:lang="en">Actually create the
classifier</tc-profile:textDescription>

  <tc-profile:hasInput rdf:resource="#Session-id"/>
  <tc-profile:hasOutput rdf:resource="#OWLS-Service"/>
</tc-profile:TC-Resource>

```

```

<!-- Process Model description -->
<process:AtomicProcess
rdf:ID="ClassifierCreation_CreateClassifier_ProcessModel">
  <service:describes
rdf:resource="#ClassifierCreation_CreateClassifier_Service"/>

  <process:hasInput rdf:resource="#Session-id"/>

```

```

        <process:hasOutput rdf:resource="#OWLS-Service"/>
</process:AtomicProcess>

<process:Input rdf:ID="Session-id">
    <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/process_
management.owl#session-id</process:parameterType>
        <rdfs:label>Session id</rdfs:label>
</process:Input>

<process:Output rdf:ID="OWLS-Service">
    <process:parameterType
rdf:datatype="&xsd:anyURI">http://www.daml.org/services/owl-
s/1.1/Service.owl#Service</process:parameterType>
        <rdfs:label>OWL-S Service</rdfs:label>
</process:Output>

<!-- Grounding description -->
<grounding:Wsd1Grounding
rdf:ID="ClassifierCreation_CreateClassifier_Grounding">
    <service:supportedBy
rdf:resource="#ClassifierCreation_CreateClassifier_Service"/>
    <grounding:hasAtomicProcessGrounding
rdf:resource="#ClassifierCreation_CreateClassifier_ProcessGrounding"/>
</grounding:Wsd1Grounding>

<grounding:Wsd1AtomicProcessGrounding
rdf:ID="ClassifierCreation_CreateClassifier_ProcessGrounding">
    <grounding:owlsProcess
rdf:resource="#ClassifierCreation_CreateClassifier_ProcessModel"/>
    <grounding:wsdlDocument>http://classakt.aktors.org:8000/ClassifiersServic
eEx/ClassifiersServiceEx.dll?Handler=GenClassifiersServiceExWSDL</grounding:wsd
lDocument>
    <grounding:wsdlOperation>
        <grounding:Wsd1OperationRef>

        <grounding:portType>urn:ClassifiersServiceExService#ClassifiersServiceExS
erviceSoap</grounding:portType>

        <grounding:operation>urn:ClassifiersServiceExService#CreateClassifier</gr
ounding:operation>
            </grounding:Wsd1OperationRef>
        </grounding:wsdlOperation>

        <grounding:wsdlInputMessage
rdf:datatype="&xsd:anyURI">urn:ClassifiersServiceExService#CreateClassifierIn<
/grounding:wsdlInputMessage>
            <grounding:wsdlInput>
                <grounding:Wsd1InputMessageMap>
                    <grounding:owlsParameter rdf:resource="#Session-id"/>

                <grounding:wsdlMessagePart>urn:ClassifiersServiceExService#nSessionID</gr
ounding:wsdlMessagePart>
                    <grounding:xsltTransformationString>
                        <![CDATA[
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:pm="http://classakt.aktors.org:8000/ontologies/process_management.owl#"
    rdf:keepNamespace=""
    pm:keepNamespace="">
        <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

```

```

        <xsl:template match="/" >
            <xsl:value-of select="rdf:RDF/pm:session-id/pm:has-session-id"/>
        </xsl:template>
</xsl:stylesheet>

    ]]>
        </grounding:xsltTransformationString>
        </grounding:WsdInputMessageMap>
    </grounding:wsdlInput>

    <grounding:wsdlOutputMessage
rdf:datatype="&xsd;#anyURI">urn:ClassifiersServiceExService#CreateClassifierOut
</grounding:wsdlOutputMessage>
    <grounding:wsdlOutput>
        <grounding:WsdOutputMessageMap>
            <grounding:owlsParameter rdf:resource="#OWLS-Service"/>

        <grounding:wsdlMessagePart>urn:ClassifiersServiceExService#return</ground
ing:wsdlMessagePart>
        <grounding:xsltTransformationString>
            <![CDATA[
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/">
        <xsl:variable name="X1" select="*" />
        <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#">
            <service:Service rdf:about="{X1}" />
        </rdf:RDF>
    </xsl:template>
</xsl:stylesheet>
            ]]>
        </grounding:xsltTransformationString>
        </grounding:WsdOutputMessageMap>
    </grounding:wsdlOutput>

</grounding:WsdAtomicProcessGrounding>

</rdf:RDF>

```

E.7 Classifier_ClassifyTemplate

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
    <!ENTITY owl "http://www.w3.org/2002/07/owl">
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
    <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
    <!ENTITY profile "http://www.daml.org/services/owl-s/1.1/Profile.owl">
    <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
    <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
    <!ENTITY pm
"http://classakt.aktors.org:8000/ontologies/process_management.owl">
    <!ENTITY tc "http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl">
]>

<rdf:RDF
    xmlns:owl=
        "&owl;#"
    xmlns:rdfs=
        "&rdfs;#"
    xmlns:rdf=
        "&rdf;#"
    xmlns:service=
        "&service;#"

```

```

xmlns:process=      "&process;#"
xmlns:profile=      "&profile;#"
xmlns:grounding=    "&grounding;#"
xmlns:pm=           "&pm;#"
xmlns:tc=           "&tc;#"
xmlns:xsd=          "&xsd;#"
xml:base=           "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/Classifier_ClassifySESSIONIDHERE.owl"
xmlns=              "http://www.classakt.aktors.org:8000/services/owl-
s/1.1/Classifier_ClassifySESSIONIDHERE.owl#"
>

<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="&service;"/>
  <owl:imports rdf:resource="&profile;"/>
  <owl:imports rdf:resource="&process;"/>
  <owl:imports rdf:resource="&grounding;"/>
  <owl:imports rdf:resource="&pm;"/>
  <owl:imports rdf:resource="&tc;"/>
</owl:Ontology>

<!-- Service description -->
<service:Service rdf:ID="Classifier_Classify_Service">
  <service:presents rdf:resource="#Classifier_Classify_Profile"/>

  <service:describedBy rdf:resource="#Classifier_Classify_ProcessModel"/>

  <service:supports rdf:resource="#Classifier_Classify_Grounding"/>
</service:Service>

<!-- Profile description -->
<profile:Profile rdf:ID="Classifier_Classify_Profile">
  <service:isPresentedBy rdf:resource="#Classifier_Classify_Service"/>

  <profile:serviceName xml:lang="en">CreateClassifier</profile:serviceName>
  <profile:textDescription xml:lang="en">Actually create the
classifier</profile:textDescription>

  <profile:hasInput rdf:resource="#Session-id"/>
  <profile:hasInput rdf:resource="#Document"/>
  <profile:hasOutput rdf:resource="#Classification"/>
</profile:Profile>

<!-- Process Model description -->
<process:AtomicProcess rdf:ID="Classifier_Classify_ProcessModel">
  <service:describes rdf:resource="#Classifier_Classify_Service"/>

  <process:hasInput rdf:resource="#Session-id"/>
  <process:hasInput rdf:resource="#Document"/>
  <process:hasOutput rdf:resource="#Classification"/>
</process:AtomicProcess>

<process:Input rdf:ID="Session-id">
  <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/process_
management.owl#session-id</process:parameterType>
  <rdfs:label>Classifier id</rdfs:label>
</process:Input>

<process:Input rdf:ID="Document">

```

```

        <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#Document</process:parameterType>
        <rdfs:label>Document</rdfs:label>
</process:Input>

<process:Output rdf:ID="Classification">
        <process:parameterType
rdf:datatype="&xsd:anyURI">http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#Classification</process:parameterType>
        <rdfs:label>Classification</rdfs:label>
</process:Output>

<!-- Grounding description -->
<grounding:WsdgGrounding rdf:ID="Classifier_Classify_Grounding">
        <service:supportedBy rdf:resource="#Classifier_Classify_Service"/>
        <grounding:hasAtomicProcessGrounding
rdf:resource="#Classifier_Classify_ProcessGrounding"/>
</grounding:WsdgGrounding>

<grounding:WsdgAtomicProcessGrounding
rdf:ID="Classifier_Classify_ProcessGrounding">
        <grounding:owlsProcess rdf:resource="#Classifier_Classify_ProcessModel"/>
        <grounding:wsdlDocument>http://classakt.aktors.org:8000/ClassifierService
/ClassifierService.dll?Handler=GenClassifierServiceWSDL</grounding:wsdlDocument
>
        <grounding:wsdlOperation>
                <grounding:WsdgOperationRef>

                <grounding:portType>urn:ClassifierServiceService#ClassifierServiceService
Soap</grounding:portType>

                <grounding:operation>urn:ClassifierServiceService#Classify</grounding:ope
ration>
                </grounding:WsdgOperationRef>
        </grounding:wsdlOperation>

        <grounding:wsdlInputMessage
rdf:datatype="&xsd:anyURI">urn:ClassifierServiceService#ClassifyIn</grounding:
wsdlInputMessage>
        <grounding:wsdlInput>
                <grounding:WsdgInputMessageMap>
                        <grounding:owlsParameter rdf:resource="#Document"/>

                        <grounding:wsdlMessagePart>urn:ClassifierServiceService#bstrInstanceText<
/grounding:wsdlMessagePart>
                                <grounding:xsltTransformationString>
                                        <![CDATA[
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:tc="http://classakt.aktors.org:8000/ontologies/TC-
ProfileHierarchy.owl#"
rdf:keepNamespace=""
tc:keepNamespace="">
        <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
        <xsl:template match="/ ">
                <xsl:value-of select="rdf:RDF/tc:Document/tc:has-location"/>
        </xsl:template>
</xsl:stylesheet>
]]>
                                </grounding:xsltTransformationString>
                        </grounding:WsdgInputMessageMap>

```

```

</grounding:wsdlInput>

<grounding:wsdlInputMessage
rdf:datatype="&xsd;#anyURI">urn:ClassifierServiceService#ClassifyIn</grounding:
wsdlInputMessage>
  <grounding:wsdlInput>
    <grounding:WsdInputMessageMap>
      <grounding:owlsParameter rdf:resource="#Session-id"/>

    <grounding:wsdlMessagePart>urn:ClassifierServiceService#nSessionID</groun
ding:wsdlMessagePart>
      <grounding:xsltTransformationString>
        <![CDATA[
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/" >
    <xsl:value-of select="SESSIONIDHERE"/>
  </xsl:template>
</xsl:stylesheet>
]]>
      </grounding:xsltTransformationString>
    </grounding:WsdInputMessageMap>
  </grounding:wsdlInput>

  <grounding:wsdlOutputMessage
rdf:datatype="&xsd;#anyURI">urn:ClassifierServiceService#ClassifyOut</grounding
:wsdlOutputMessage>
    <grounding:wsdlOutput>
      <grounding:WsdOutputMessageMap>
        <grounding:owlsParameter rdf:resource="#Classification"/>

      <grounding:wsdlMessagePart>urn:ClassifierServiceService#classes</groundin
g:wsdlMessagePart>
        <grounding:xsltTransformationString>
          <![CDATA[
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:ns="urn:ClassifierServiceService">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="classes">
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:tc-ont="http://classakt.aktors.org/ontologies/TC-ProfileHierarchy#">
      <xsl:for-each select="//ns:Classed">
        <tc-ont:Classification>
          <tc-ont:has-ref><xsl:apply-templates
select="ref"/></tc-ont:has-ref>
          <tc-ont:has-title><xsl:apply-templates
select="title"/></tc-ont:has-title>
        </tc-ont:Classification>
      </xsl:for-each>
    </rdf:RDF>
  </xsl:template>
</xsl:stylesheet>
]]>
        </grounding:xsltTransformationString>
      </grounding:WsdOutputMessageMap>
    </grounding:wsdlOutput>

  </grounding:WsdAtomicProcessGrounding>

</rdf:RDF>

```

Glossary

Machine Learning

Association for Computing Machinery (ACM) – Organisation which advances Computer Science through its on-line portal, publications and conferences.

Bag of words – Document representation consisting of a list of words and the number of times the words appear (term frequency).

Feature – A word or word stem used for text classification. Typically the set of features is a select subset of the set of words which are to be found in a document set.

Naïve Bayes Classifier – A probabilistic classifier based on Bayes Theorem. Term probabilities are assigned to classes and for a new document the probability of belonging to any particular class is computed.

Stemming – Removal of suffixes from words in order to reduce the number of terms that are synonyms in a textual document. A popular algorithm is the Porter Stemming Algorithm.

Stopword – A common word the frequency of which is evenly distributed across all textual documents so that it may thus be safely excluded from feature sets used for text classification.

Support Vector Machine (SVM) – A machine learning system based on recent advances in statistical learning theory and adapted with promising results for use in text classifiers.

Taxonomy – A classification of some things, often in the form of a hierarchy.

Weka library – A collection of machine learning algorithms written in Java for data mining made available by the University of Waikato.

Semantic Web

Grid - the development and advancement of technologies that provide seamless and scalable access to wide-area distributed resources. Computational Grids enable the sharing, selection, and aggregation of a wide variety of geographically distributed computational resources (such as supercomputers).

Inputs, Outputs, Preconditions and Effects (IOPEs) – Part of the functional description of a Semantic Web service describing a service's inputs, outputs and conditions which must hold before and after service execution.

Multipurpose Internet Mail Extensions (MIME) – Extends the format of Internet mail to allow non-US ASCII textual messages, non-textual messages, multipart message bodies and non-US ASCII information in message headers.

Ontology – A specification of a conceptualization. A model of the concepts and relationships which exist in a system.

OWL-S – An OWL based Web service ontology. An initiative of the OWL-S Coalition which in addition to the service ontology works to support tools and agent technology to enable automation of services on the Semantic Web

RDF Schema (RDFS) – A semantic extension of RDF to provide a vocabulary description language for RDF. That is, it is a simple ontology language.

Resource Description Framework (RDF) – A lightweight ontology system with an XML serialization designed to support the exchange of knowledge on the Web.

Semantic Grid - an extension of the current Grid incorporating Semantic Web technologies in which information and services are given well-defined meaning, better enabling computers and people to work in cooperation.

Web Ontology Language (OWL) – Language designed on top of RDF and provide additional semantics within its essential role of being a machine processable language for the Web.

Web Service Modeling Ontology (WSMO) – An ontology that describes various aspects related to Semantic Web Services. An initiative of the SDK WSMO working group which aligns research efforts between the SEKT, DIP and Knowledge Web research projects.

Bibliography

- [Adams, 2001] Adams, K. C. (2001), *Word Wranglers: Automatic classification tools transform enterprise documents from "bags of words" into knowledge resources*. Intelligent KM, January 2001.
- [Baader et al., 2002] Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P. F. (2002), *The Description Logic Handbook*. Cambridge University Press.
- [Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J. and Lassila, O. (2001), *The Semantic Web*. Scientific American, 284(5):34-43, 2001.
- [Cabral et al., 2004] Cabral, L., Domingue, J., Motta, E., Payne, T. and Hakimpour, F. (2004), *Approaches to Semantic Web Services: An Overview and Comparisons*. Proceedings of the First European Semantic Web Symposium (ESWS2004); 10-12 May 2004, Heraklion, Crete, Greece.
- [Calvanese et al., 2002] Calvanese, D., De Giacomo, G. and Lenzerini, M. (2002), *Description Logics: Foundations for Class-based Knowledge Representation*. Proceedings of the 17th IEEE Symposium on Logic in Computer Science.

- [CCSDS, 2002] CCSDS (2002), *Reference Model for an Open Archival Information System (OAIS)*. CCSDS 650.0-B-1 Blue Book (Standard). Issue 1. January 2002.
- [CCSDS, 2004] CCSDS (2004), *Producer - Archive Interface Methodology Abstract Standard*. CCSDS 651.0-B-1 Blue Book (Standard). Issue 1. May 2004.
- [Chang & Lin, 2004] Chang, C.-C. and Lin, C.-J. (2004), *LIBSVM: a Library for Support Vector Machines*. Technical Report, see www.csie.ntu.edu.tw/~cjlin/libsvm/.
- [Chen et al., 2003] Chen, P.-H., Lin, C.-J. and Schölkopf, B. (2003), *A Tutorial on ?-Support Vector Machines*. Available via www.kernel-machines.org/tutorial.html.
- [Cortes & Vapnik, 1995] Cortes, C. and Vapnik, V. (1995), *Support-Vector Networks*. Machine Learning, v.20 n.3, p273-297, Sept 1995.
- [CPA, 1996] CPA (1996), *Preserving Digital Information: Report of the Task Force on Archiving of Digital Information*. Commission on Preservation and Access May 1996. Available www.rlg.org/ArchTF/.
- [Cristianini & Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000), *An Introduction to Support Vector Machines*. Cambridge University Press.
- [Crubézy et al., 2001] Crubézy, M., Lu, W., Motta, E. and Musen, M. A. (2001), *The Internet Reasoning Service: Delivering Configurable Problem-Solving Components to Web Users*. K-CAP 2001 Workshop, Canada.
- [De Roure et al., 2005] De Roure, D., Jennings, N. R. and Shadbolt, N. R. (2005), *The Semantic Grid: Past, Present, and Future*. Proceedings of the IEEE, Vol. 93, No.3, March 2005.

- [Doerr et al., 2003] Doerr, M., Hunter, J. and Lagoze, C. (2003), *Towards a Core Ontology for Information Integration*. Journal of Digital Information, Volume 4, Issue 1.
- [Duda & Hart, 1973] Duda, R. and Hart, P. (1973), *Pattern Classification and Scene Analysis*. Wiley, NY.
- [Dumais et al., 1998] Dumais, S., Platt, J., Heckerman, D. and Sahami, M. (1998), *Inductive Learning Algorithms and Representations for Text Categorization*. Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management, pages 148-155.
- [Eyheramendy et al., 2003] Eyheramendy, S., Lewis, D. D. and Madigan, D. (2003), *On The Naïve Bayes Model for Text Categorization*. Artificial Intelligence and Statistics 2003.
- [Fensel & Bussler, 2002] Fensel, D. and Bussler, C. (2002), *The Web Service Modeling Framework WSMF*. Electronic Commerce: Research and Applications, 1 (2002) 113-137.
- [Fensel & Motta, 1998] Fensel, D. and Motta, E. (1998), *Structured Development of Problem Solving Methods*. Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW'98).
- [Gilliland-Swetland, 2000] Gilliland-Swetland, A. J. (2000), *Enduring Paradigm. New Opportunities: The Value of the Archival Perspective in the Digital Environment*. Council on Library and Information Resources, 2000.
- [Goble & De Roure, 2004] Goble, C. and De Roure, D. (2004), *The Semantic Grid: Myth Busting and Bridge Building*. 16th European Conference on Artificial Intelligence (ECAI-2004), pages 1129-1135, Valencia, Spain, 2004.

- [Harris & Gibbins, 2003] Harris, S. and Gibbins, N. (2003), *3store: Efficient Bulk RDF Storage*. Proceedings 1st International Workshop on Practical and Scalable Semantic Web Systems, Sanibel Island, Florida, USA.
- [Heflin & Hendler, 2001] Heflin, J. and Hendler, J. (2001), *A Portrait of the Semantic Web in Action*. IEEE Intelligent Systems, 16(2):54-59.
- [Horrocks, 2002] Horrocks, I. (2002), *DAML + OIL: A Description Logic for the Semantic Web*. IEEE Bull. On Data Engineering, 25(1):4-9, 2002.
- [Hsu et al., 2003] Hsu, C.-W., Chang, C.-C. and Lin, C.-J. (2003), *A Practical Guide to Support Vector Classification*. Technical Document from Department of Computer Science and Information Engineering, National Taiwan University.
- [Hsu & Lin, 2002] Hsu, C.-W. and Lin, C.-J. (2002), *A Comparison of Methods for Multi-class Support Vector Machines*. IEEE Transactions on Neural Networks, 13(2):415-425, March 2002.
- [Hunter, 2003] Hunter, J. (2003), *Enhancing the Semantic Interoperability of Multimedia through a Core Ontology*. IEEE Transactions on Circuits and Systems for Video Technology, January, 2003.
- [IBM, 2003] IBM (2003), *Specification: Business Process Execution Language for Web Services Version 1.1*. Available at www-106.ibm.com/developerworks/library/ws-bpel/.
- [ISO, 1986] ISO (1986), *Information processing -- Text and office systems -- Standard Generalized Markup Language (SGML)*. ISO 8879:1986 available at <http://www.iso.org>.

- [Joachims, 1998] Joachims, T. (1998), *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. Proceedings of ECML-98, 10th European Conference on Machine Learning (Chemnitz, DE, 1998), pp 4-15.
- [John & Langley, 1995] John, G. H. and Langley, P. (1995), *Estimating Continuous Distributions in Bayesian Classifiers*. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence.
- [Kalfoglou & Schorlemmer, 2003] Kalfoglou, Y. and Schorlemmer, M. (2003), *Ontology Mapping: the state of the art*. The Knowledge Engineering Review 18(1).
- [Klein & Potter, 2004] Klein, E. and Potter, S. (2004), *An Ontology for NLP Services*. Proceedings of Workshop on a Registry of Linguistic Data Categories within an Integrated Language Resource Repository Area, LREC 2004, May 2004.
- [Lagoze & Hunter, 2001] Lagoze, C. and Hunter, J. (2001), *The ABC Ontology and Model*. Journal of Digital Information, 2 (2), 2001.
- [Lagoze & van de Sompel, 2001] Lagoze, C. and van de Sompel, H. (2001), *The Open Archives Initiative Protocol for Metadata Harvesting*. Open Archives Initiative, January 2001. Available
<<http://www.openarchives.org/OAI/openarchivesprotocol.htm>>.
- [Lara et al., 2003] Lara, R., Lausen, H., Arroyo, S., Bruijn, J. d. and Fensel, D. (2003), *Semantic Web Services: description requirements and current technologies*. International Workshop on Electronic Commerce, Agents, and Semantic Web Services, In conjunction with the Fifth International Conference on Electronic Commerce (ICEC 2003).

- [Lewis, 1991] Lewis, D. D. (1991), *Evaluating Text Categorization*. Proc. Speech and Natural Language Workshop pp 312-318, 1991.
- [Lewis, 1998] Lewis, D. D. (1998), *Naïve (Bayes) at forty: The independence assumption in information retrieval*. Proceedings of ECML-98, 10th European Conference on Machine Learning (Chemnitz, DE, 1998), pp 4-15.
- [Maedche et al., 2002] Maedche, A., Motik, B., Silva, N. and Volz, R. (2002), *MAFRA - A Mapping FRamework for Distributed Ontologies*. Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW.
- [Mandell & McIlraith, 2003] Mandell, D. J. and McIlraith, S. A. (2003), *A Bottom-Up Approach to Automating Web Service Discovery, Customization, and Semantic Translation*. Proceedings of the Twelfth International World Wide Web Conference Workshop on E-Services and the Semantic Web (ESSW '03). Budapest, 2003.
- [Martin et al., 2004] Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N. and Sycara, K. (2004), *Bringing Semantics to Web Services: The OWL-S Approach*. Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), July 6-9, 2004, San Diego, California, USA.
- [Martin et al., 1999] Martin, D. L., Cheyer, A. J. and Moran, D. B. (1999), *The Open Agent Architecture: A Framework for Building Distributed Software Systems*. Applied Artificial Intelligence, vol. 13, no. 1-2, pp. 91-128, January-March 1999.
- [McCallum & Nigam, 1998] McCallum, A. and Nigam, K. (1998), *A Comparison of Event Models for Naïve Bayes Text Classification*. AAAI-98 Workshop on Learning for Text Categorization.

- [McIlraith & Son, 2002] McIlraith, S. and Son, T. C. (2002), *Adapting Golog for Composition of Semantic Web Services*. Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002), April 2002.
- [McIlraith et al., 2001] McIlraith, S., Son, T. C. and Zeng, H. (2001), *Semantic Web Services*. IEEE Intelligent Systems, 16(2):46-53, 2001.
- [Mitchell, 1997] Mitchell, T. M. (1997), *Does Machine Learning Really Work?* AI Magazine September 22, 1997.
- [Motta et al., 2003] Motta, E., Domingue, J., Cabral, L. and Gaspari, M. (2003), *IRS-II: A Framework and Infrastructure for Semantic Web Services*. 2nd International Semantic Web Conference (ISWC2003) 20-23 October 2003, Sundial Resort, Sanibel Island, Florida, USA.
- [OWL-S, 2004] OWL-S (2004), *OWL-S: Semantic Markup for Web Services*. The OWL Services Coalition see <http://www.daml.org/services/owl-s/1.0/owl-s.pdf>.
- [OWL-S, 2005] OWL-S (2005), *OWL-S 1.1*. The OWL-S Services Coalition see <http://www.daml.org/services/owl-s/1.1/>.
- [Paolucci et al., 2002a] Paolucci, M., Kawamura, T., Payne, T. and Sycara, K. (2002a), *Semantic Matching of Web Services Capabilities*. Proceedings of International Semantic Web Conference (ISWC), Sardinia, Italy, 2002.
- [Paolucci et al., 2002b] Paolucci, M., Kawamura, T., Payne, T. and Sycara, K. (2002b), *Importing the Semantic Web in UDDI*. Proceedings of E-Services and the Semantic Web (ESSW02).
- [Platt, 1998] Platt, J. C. (1998), *Sequential Minimum Optimization: A Fast Algorithm for Training Support Vector Machines*. MSR-TR-98-14, Microsoft Research, 1998.

- [Platt et al., 2000] Platt, J. C., Cristianini, N. and Shawe-Taylor, J. (2000), *Large Margin DAGs for Multiclass Classification*. Advances in Neural Information Processing Systems 12, 2000.
- [Rogati & Yang, 2002] Rogati, M. and Yang, Y. (2002), *High-Performing Feature Selection for Text Classification*. CIKM'02.
- [Sabou et al., 2003] Sabou, M., Richards, D. and van Splunter, S. (2003), *An experience report on using DAML-S*. Proceedings of WWW 2003 Workshop on E-Services and the Semantic Web (ESSW'03).
- [Schölkopf, 2000] Schölkopf, B. (2000), *Statistical Learning and Kernel Methods*. MSR-TR 2000-23, Microsoft Research.
- [Sebastiani, 1999] Sebastiani, F. (1999), *Machine Learning in Automated Text Classification*. Technical Report B4-31 Istituto di Elaborazioni dell'Informazione, Consiglio delle Ricerche, Italy.
- [Smith et al., 2003] Smith, M., Barton, M., Bass, M., Branschovsky, M., McClellan, G., Stuve, D., Tansley, R. and Harford-Walker, J. (2003), *DSpace: An Open Source Dynamic Digital Repository*. D-Lib Magazine, Volume 9 Number 1, Jan 2003.
- [UDDI.org, 2000] UDDI.org (2000), *UDDI Technical White Paper*. Available at <http://www.uddi.org/whitepapers.html>.
- [van der Aalst, 2003] van der Aalst, W. M. P. (2003), *Don't go with the flow: Web services composition standards exposed*. IEEE Intelligent Systems, Jan/Feb 2003.
- [W3C, 1999a] W3C (1999a), *XML Path Language (XPath)*. <http://www.w3.org/TR/1999/REC-xpath-19991116>.

- [W3C, 1999b] W3C (1999b), *XSL Transformations (XSLT)*.
<http://www.w3.org/TR/1999/REC-xslt-19991116>.
- [W3C, 2000a] W3C (2000a), *XML Linking Language (XLink) Version 1.0*.
<http://www.w3.org/TR/2000/REC-xlink-20010627/>.
- [W3C, 2000b] W3C (2000b), *Simple Object Access Protocol (SOAP) 1.1*.
<http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>.
- [W3C, 2001a] W3C (2001a), *XML Schema Part 0: Primer*.
<http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/>.
- [W3C, 2001b] W3C (2001b), *XML Pointer Language (XPointer) Version 1.0*.
<http://www.w3.org/TR/2001/WD-xptr-20010108>.
- [W3C, 2001c] W3C (2001c), *Web Services Description Language (WSDL) 1.1*.
<http://www.w3.org/TR/2001/NOTE-wsdl-20010315>.
- [W3C, 2004a] W3C (2004a), *Extensible Markup Language (XML) 1.0 Third Edition*.
<http://www.w3.org/TR/2004/REC-xml-20040204>.
- [W3C, 2004b] W3C (2004b), *Namespaces in XML 1.1*.
<http://www.w3.org/TR/2004/REC-xml-names11-20040204>.
- [W3C, 2004c] W3C (2004c), *RDF Primer*. <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.
- [W3C, 2004d] W3C (2004d), *RDF Vocabulary Description Language 1.0: RDF Schema*.
<http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>.
- [W3C, 2004e] W3C (2004e), *OWL Web Ontology Language Reference*.
<http://www.w3.org/TR/2004/REC-owl-guide-20040210/>.

- [W3C, 2005] W3C (2005), *Representing Classes As Property Values on the Semantic Web*. <http://www.w3.org/TR/2005/NOTE-swbp-classes-as-values-20050405>.
- [Weston & Watkins, 1999] Weston, J. and Watkins, C. (1999), *Multi-class Support Vector Machines*. Proceedings of ESANN99, Brussels.
- [Wohed et al., 2002] Wohed, P., van der Aalst, W. M. P., Dumas, M. and ter Hofstede, A. H. M. (2002), *Pattern Based Analysis of BPEL4WS*. QUT Technical report, FIT-TR-2002-05, Queensland University of Technology, Queensland University of Technology, Brisbane.
- [WSMO, 2004] WSMO (2004), *D2v1.1. Web Service Modeling Ontology*. SDK WSMO Working Group available at <http://www.wsmo.org/2004/d2/v0.2/20040306/>.
- [WSMX, 2004a] WSMX (2004a), *D13.1v0.1 Web Service Modeling Execution Environment - Conceptual Model (WSMX_O)*. SDK WSMX Working Group available at <http://www.wsmo.org/2004/d13/d13.1/v0.1/>.
- [WSMX, 2004b] WSMX (2004b), *D13.4v0.1 WSMX Architecture*. SDK WSMX Working Group available at <http://www.wsmo.org/2004/d13/d13.4/v0.1/20040622>.