TRAINING-LESS ONTOLOGY-BASED TEXT CATEGORIZATION

by

MACIEJ JANIK

(Under the Direction of Krzysztof Kochut)

ABSTRACT

In this dissertation, we present a novel approach for automatic text categorization that combines some features of traditional text categorization approaches with the knowledge discovery methods in the Semantic Web. The proposed method, OmniCat, uses ontological knowledge to perform categorization of text documents. More specifically, it concentrates on the recognized named entities and relationships in document text to perform the categorization according to the taxonomy in the used ontology.

Traditional approaches in text categorization use machine learning to train a classifier on a pre-classified set of training documents, in order to learn the distinguishing features of each category. Later, such classifier is used to classify previously unseen documents. One of the distinguishing features of OmniCat is that, unlike in the traditional text categorization methods, it does not rely on the existence of a training set and only uses formal knowledge from the ontology to perform categorization. In the proposed approach, the ontology effectively becomes the classifier. This makes OmniCat a training-less ontology-based text categorization method.

A classification category in OmniCat is defined as an ontology fragment, and can be seen as a context of interest for categorization. Categorization in this model depends on finding the best semantic fit of the text document that is transformed to a semantic graph, into one of the defined contexts. Contexts can be dynamically redefined, as the user's interest changes. Another distinguishing feature of OmniCat is that neither re-training of the classifier, nor supplying a new training set of documents is needed with the change of classification contexts. Simply, OmniCat classifies documents according to the newly supplied contexts of interest.

The results from the performed experiments are highly encouraging. OmniCat can achieve a high categorization accuracy, which is within the range of the other tested traditional text categorization methods. We believe that as more rich and comprehensive domain ontologies become available, the proposed approach offers a suitable alternative to the traditional text categorization methods.

The implementation of the proposed categorization method builds upon two necessary components, also created as part of this dissertation. The first is BRAHMS, a high-performance main-memory ontology storage engine, which provides an extensive and very fast API for querying large ontologies. The second is SPARQLeR, an extension of the SPARQL query language with regular path expressions for the semantic association discovery. The algorithms and querying methods developed for SPARQLeR became an integral part of the proposed ontology-based categorization.

In the dissertation, we describe OmniCat, which is the main contribution of this dissertation, and both components, BRAHMS and SPARQLeR, which are the integral parts of the created prototype. We provide a detailed description of each system and the results of the performed experiments.

INDEX WORDS: Text Categorization, Ontology-based Text Categorization, Semantic Web, Ontology, Semantic Associations

TRAINING-LESS ONTOLOGY-BASED TEXT CATEGRIZATION

by

MACIEJ JANIK

Master of Science, AGH University of Science and Technology, Krakow, Poland, 1999

Master of Science, AGH University of Science and Technology, Krakow, Poland 2001

A Dissertation Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2008

TRAINING-LESS ONTOLOGY-BASED TEXT CATEGORIZATION

by

MACIEJ JANIK

Major Professor:	Krzysztof J. Kochut
Committee:	John A. Miller
	Khaled Rasheed
	Amit P. Sheth

DEDICATION

I dedicate this dissertation to my beloved family – to my parents, as without their love and support over the years this work would not be possible, and to my loving wife, Małgosia, who makes our life an inspirational journey.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Krzysztof J. Kochut for his guidance, advice, support, and the inspirational discussions we had. I thank him for being very generous with his time and wisdom. I would also like to thank members of my doctoral committee, Professors John A. Miller, Kahled Rasheed, and Amit P. Sheth for their valuable suggestions and guidance.

I have been lucky to be a part of a group of enthusiastic young researchers at the LSDIS lab. I appreciate the opportunities for collaborations and discussions with variety of people, including Cartic Ramakrishnan, Boanerges Aleman-Meza, Matthew Eavenson, Matt Perry, and other members and alumni of the LSDIS lab. Opportunity to work with them and learn from them has been invaluable.

Finally, I would like to thank my wife, Magłosia, for her support over the years and encouragement for continuing the academic career.

TABLE OF CONTENTS

LIST OF FIGURES

Page

CHAPTER 1 INTRODUCTION

Text categorization is a task of assigning one or more predefined categories to an electronic text document, based on its content. It has been an active field of research for many years and multiple methods have been proposed and developed. There are two basic types of categorizations: supervised and unsupervised. All of the supervised categorization methods share the same property – they use machine learning techniques to train the classifier using a provided training set of pre-classified documents to later perform the categorization of previously unseen documents. During such training, a classifier gathers knowledge that is essential for distinguishing categories based on document features. Unsupervised methods do not use any kind of external information for categorization. They focus on discovering such features of documents that enable them to assemble similar documents into coherent groups, usually called clusters. In this dissertation, we will focus only on supervised text categorization.

The Semantic Web [22] and ontologies have been present in Computer Science for almost a decade. Ontologies contain formal descriptions of concepts and relationships that model certain domains. An ontology defines knowledge and properties of a given domain in a way that machines can read it and understand it.

In this dissertation, we propose applying ontological knowledge for the text categorization purposes. In contrast to previously taken approaches in text categorization that enhance known methods with ontological knowledge, we propose to directly use ontological knowledge for text categorization. The novelty of our method is that it does not rely on the training of a categorizer, making a training set unnecessary, and directly leveraging the knowledge from the ontology for text categorization. We argue that instead of learning the distinguishing features of each category from documents in the training set, we are able to extract specific knowledge about the interesting domain or category from the ontology and use it for categorizing documents.

Not requiring a training set and providing category definitions by relevant ontology fragments, allows for much greater flexibility than in the traditional text categorization methods. Categories recognized by the categorization method are no longer limited by the availability of training documents. Furthermore, the user can build dynamic contexts of interests for categorization, and while using the same ontology and the same document corpora, classify documents in a variety of ways according to different sets of interesting features.

1.1. Contributions

The contributions of this dissertation include a novel approach for using ontologies for the automatic text categorization and two advanced software subsystems that made the approach feasible. The method presented in this dissertation for training-less ontology-based text categorization combines some features of traditional text categorization with approaches specific to the area of Semantic Web. We introduce a new methodology and algorithms for text categorization using ontologies. To some extent, the new methodology relies on the know-how and solutions of already known approaches from the area of text categorization, natural language processing, Semantic Web, and knowledge management.

More specifically, the contributions of this dissertation are as follows:

(1) BRAHMS [52], a high-performance main-memory ontology storage engine optimized for efficient querying of large ontologies. BRAHMS provides the foundation for the overall

categorization system, as it enables a highly efficient execution of complex ontology queries needed for the ontology-based categorization tasks. The utilization of BRAHMS makes the proposed categorization method efficient enough for practical applications.

(2) SPARQLeR [63], an extension of the SPARQL query language with variable-length path queries. Graph patterns in SPARQLeR may involve path patterns, defined with the use of regular expressions on properties and certain conditions on their elements. This extension enables efficient discovery of semantic associations that have certain desired features, even in large ontologies. SPARQLeR was originally used in the Glycomics project [4] for discovering metabolic pathways that can be interpreted as long semantic associations. Its applications are not only limited to bio-informatics, as it has been successfully employed in the proposed text categorization method.

(3) OmniCat [53] [54] [55], an ontology-based automatic text categorization method that does not require a training set and allows dynamic definition of classification categories based on the ontology schema. The novelty of this approach is that it does not depend on the existence of a training set. Instead of training a classifier on the pre-classified documents, it leverages the knowledge from a domain ontology, including entities, relationships, and the class taxonomy, to categorize a text document. Effectively, a category is defined by selected ontology fragments. Consequently, this approach allows for independently defining multiple categories and treating them as different contexts of interest. Selecting specific contexts for use in categorization changes the perspective through which the document is perceived. Therefore, the documents can be very easily reclassified according to the user's changing interests.

The proposed method requires a transformation of the unstructured text into a semantic graph, which is performed in the first phase of the categorization. The transformation is achieved by utilizing already known methods such as named entity recognition, entity disambiguation and relationship identification. The second phase utilizes graph similarity measures to find a fragment of the ontology that best matches the document's semantic graph. The used similarity measures incorporate both structural and semantic properties of the created graph. Selecting the best matching ontology fragment, based on the calculated similarity, determines the ontological domain that provides the best description of the document. The categorization phase finds taxonomy classes in the ontology that best describe the recognized domain and matches them with the user-defined categories of interest. Alternatively, the user defined categories are used for matching. The categorization system is built with the use of BRAHMS, the previously introduced main-memory ontology engine, and utilizes SPARQLeR, the created language extensions for finding taxonomy class coverage. We describe the implemented prototype of the system and present its evaluation, including a comparison with selected traditional text categorization methods.

1.2. Context and scope

Automatic text categorization has been one of the central interests of the information retrieval community for many years and intensive research has been conducted in this area. Nowadays, as large amounts of electronic documents are available, automatic text categorization became a necessity and is extensively used in many contexts. One of the examples is the automatic classification of incoming electronic news into categories, such as entertainment, politics, business, sports, and so on. Traditional approaches to supervised categorization utilize statistical or machine learning methods to perform this task. All of these methods require a training set of pre-classified documents that is used for classifier training in order to learn the characteristics of one or more categories; a comprehensive survey of text categorization methods can be found in [96]. Later, based on the extracted features, the classifier can be used to assign categories to previously unseen documents.

The use of external knowledge, semantics and named entities for the categorization purposes has been addressed by the information retrieval community. Most approaches utilized an external dictionary, such as WordNet [8], to unify the vocabulary, find synonyms and group words to reduce the dimensionality and improve the categorization accuracy. Ontologies have much more to offer for the benefit of text categorization. Knowledge is encapsulated into formal structures that have specific semantics. Such semantics and structural information can be used for various purposes during the categorization process. Some approaches include the named entity neighborhood and link analysis, while others exploit taxonomical relationships. All these efforts have incorporated named entities and semantics into the categorization process to achieve better results. However, all of the proposed solutions are based on the assumption of the existence of a training set of pre-classified documents and learning the categories' features from them. The method proposed in this dissertation moves away from learning the categories from the documents towards defining the categories in terms of ontological classes, instances, relationships, and ontology fragments. It also relinquishes the requirement of a training set, which is a novel approach in the text categorization field. Instead, the proposed method exploits the semantics of the recognized named entities and knowledge gathered in the domain ontology and attempts to find the best matching categories in the ontology schema. This allows a document to be classified *by* the ontology and *into* the ontology. The ontological knowledge is used to recognize entities, which determine the final categorization. The document, by the category assignment of its entities, is later categorized into the taxonomy defined in the ontology.

The research challenges in dealing with the use of ontology for the training-less automatic text categorization include problems both from the Information Retrieval and Semantic Web systems areas. Ontologies containing millions of entities are more common nowadays [102]. Some of the key challenges for the proposed categorization scheme are in handling and efficient querying of such large ontologies. The techniques presented in this dissertation extensively use graph traversals and path queries to determine how entities are connected in the ontology and to find the most important entities among them in a given context. The implementation of an efficient ontology storage engine and a suitable query language with the necessary operators are among several different solutions presented in this dissertation. Additionally, a transformation of unstructured text into a semantic graph poses challenges related to named-entities, their recognition in the text, proper name disambiguation, and the relationship extraction. The use of the created semantic graph for text-categorization based on the ontological knowledge required creating hybrid metrics that combine graph similarity with text categorization. Finally, other challenges exist in the evaluation of the proposed categorization method. The ontology-based categorization assigns classes from the ontology's schema taxonomy that best describe the document content, whereas user defined categories can span multiple classes and domains. The evaluation must include both the quality of the mapping of the ontological knowledge into the user category definitions that capture the user's perceived interest as well as the ontological class assignment, based on the document content and the knowledge represented in the ontology.

The challenges involved in the training-less ontology-based text categorization involve research in multiple areas spanning the intersection of text categorization and Semantic Web. The proposed methods and solutions are rooted in already well-established approaches. This dissertation presents a novel methodology for text categorization, where categories can be defined dynamically by ontological contexts and the lack of a training set is no longer an obstacle for performing categorization. Even multiple categorizations of the same document can be performed using different perspectives and interests without modifying the categorizer or the ontology.

In this dissertation, we also describe two separately created systems that made the training-less ontology-based text categorization possible. Significant new contributions in the area of Semantic Web that were used as building blocks for the OmniCat categorization method include BRAHMS, a high-performance main-memory storage engine for RDF/S ontology, and SPARQLeR, an extension of SPARQL language with path regular expressions.

CHAPTER 2 BACKGROUND AND RELATED WORK

This chapter describes the background information on methods and components which are highly relevant for the proposed method of ontology-based text categorization. The first part of the chapter is related to the notion of semantics, and includes definitions and background information about the Semantic Web, ontologies, ontology storage, and the available query languages. The second part of the chapter focuses on presenting the related work in the area of the automatic text categorization and the use of external knowledge for this task. In the text categorization method proposed in this dissertation, we rely on the external knowledge in the form of an ontology. Finally, we provide some general information about natural language processing (NLP) techniques that are applicable for our training-less ontology-based text categorization method.

2.1. Semantic Web

The Semantic Web [22] is an extension of the current World Wide Web [21], that aims to introduce a level of semantics to the available information and resources, so that they can become more easily accessible and understandable by computers. The initial version of the Web was designed in a way that humans could easily understand the content of the pages and navigate among them, despite of different page layouts, used images, and presentation of hyperlinks or buttons. Computers, on the other hand, have very limited capability of processing and understanding of unstructured information. The Semantic Web provides a common framework for defining and sharing the explicit semantics of the presented information in a machine-readable form. It enables computers to perform more complicated tasks that involve searching, managing and combining semantically annotated information by utilizing domain-defined information with the support of a formal reasoning model.

The Resource Description Framework (RDF) [47] is the formal specification of semantics in a machine-consumable form. It is a model that allows capturing the semantics of entities by making explicit statements about them, their properties and relationships to other resources or class of resources. Additional languages that support different scopes of formal reasoning models are built on top of RDF. These include Resource Description Framework Schema (RDFS) [24], and Ontology Web Languages (OWL): OWL-Lite, OWL-DL, and OWL-Full [80]. Each of the presented languages has a different flavor of logic used for reasoning about the original statements and a different expressive power. RDFS, in addition to the RDF vocabulary, provides a definition of schema class, domain and range properties, and subsumption in taxonomy of classes and properties. Semantics of OWL-Light and OWL-DL languages can be mapped respectively to SHIF(D) and SHOIN(D) description logic [50]. Finally, OWL-Full, as the most expressive language, goes beyond the description logic and can be modeled with the first order logic [79], or the first order logic with reified predicates [120]. The proposed categorization methodology relies only on the logic provided by RDF/RDFS model.

2.2. Ontology

An Ontology is a “specification of a conceptualization” [34]. Semantic Web applications use it to process and understand information from specific domains that it models. Ontology provides a description of a domain on two levels: schema and instances. The schema includes a description of the meta-information about a domain, including classes typically arranged into a hierarchy and relationships between them that may have specific logic properties (e.g. transitivity, symmetry, and others). It describes the domain on a conceptual level that enables a proper interpretation of the underlying instance entities and structures. The interconnected instances provide the factual knowledge about the domain.

Ontology may be represented using a triple-based data model, where a statement is represented by a triple (subject, predicate, object). A statement comprises of a single fact from the knowledge, and describes how the two given named entities are related. An instance base is composed of factual knowledge about entities and their relationships, while meta-information describing concepts, properties and taxonomical hierarchy is included in the schema. Ontology can also be represented using a directed labeled graph, where entities are nodes and properties are the labeled edges connecting them. This interpretation allows using of a well-developed graph theory to help solve problems that utilize ontologies. A sample ontology fragment with schema and instances is presented in Figure 2.1.

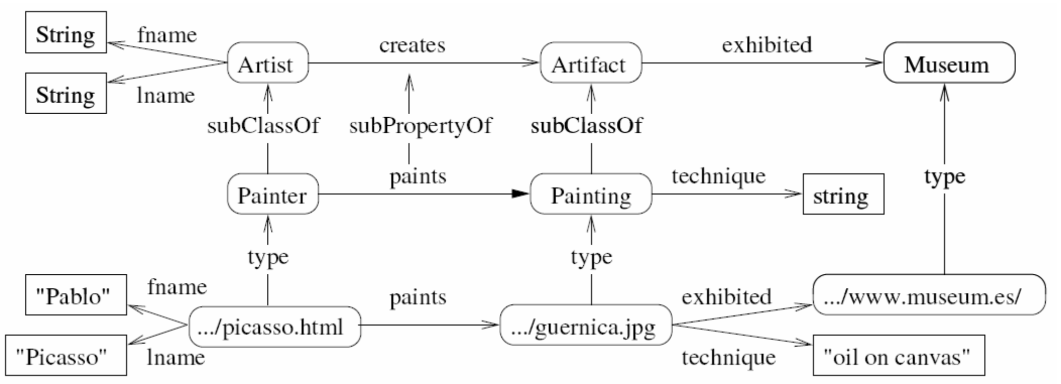


Figure 2.1 Fragment of an ontology for Cultural Portal [15]

Well-populated ontologies are essential for many Semantic Web applications. They are at the core of most operations that involve semantics, such as reasoning, semantic disambiguation and categorization, or knowledge discovery. Applications may focus on different important features of the ontology. For some tasks, such as entity recognition, the coverage and the level of detail is crucial. Ontology coverage describes if a given ontology defines concepts and entities from the entire domain, or only from some parts of it. The precision measure provides explanation on how well the concepts in the ontology resemble concepts in the domain. More detailed studies and explanations for coverage and precision are provided in [35]. A good ontology for the entity recognition purpose should include a large number of named entities from the analyzed domain accompanied with possible variations of the name labels. Applications that focus on reasoning favor a well-developed schema with a proper concept hierarchy and well-defined relationships. In all cases, the coverage and precision are of high importance. An illustrative example of ontology coverage and precision is presented in Figure 2.2.

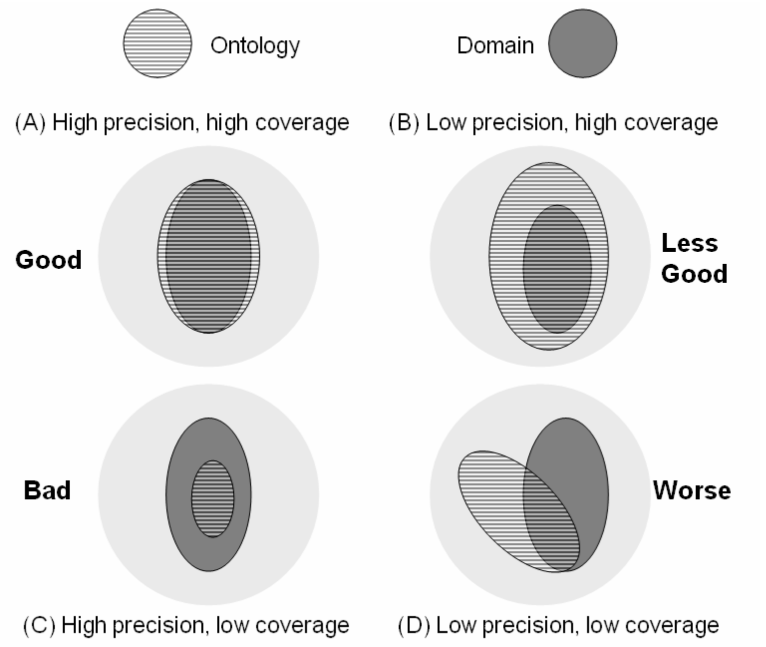

Ontology
Domain
(A) High precision, high coverage
(B) Low precision, high coverage
Good
Less Good
Bad
Worse
(C) High precision, low coverage
(D) Low precision, low coverage

Figure 2.2 Coverage and precision in ontology

All of the features mentioned above contribute to the ontology quality. However, the ontology quality is still yet to be formally defined. For each application, it may have a different meaning, as the requirements stated for different ontologies vary. Most of the structural and logical aspects of the ontology can be measured and quantified, including entity connectivity, class coverage, or logical consistency. Studies of the measures can be found in [109], [36], [107]. For the ontology-based text categorization purposes, one of the important, non-structural features is a variety of descriptive literals for named entities in a given language. Unfortunately, the quality of this feature is still hard to measure directly, and thus the question if an ontology is good for text categorization purposes is not easy to answer. Some efforts to measure the linguistic quality of an ontology have been presented in [117].

Typically, constructing ontologies for some domains require a significant human effort and time. In most cases, they are built for specific cases and domains, such as biology (GO [3], GlycO and EnzyO [91]), culture [94] or e-government [103]. Such deep ontologies concentrate on defining schema-level concepts with relationships that are highly refined, in most cases by the process of curation and verification by experts, but in general their construction and development requires a significant effort [99]. On the other side of the spectrum are shallow ontologies that are automatically created with use of extraction and annotation tools, such as presented in [42], [43], [115] and [105]. A more comprehensive survey of web extraction tools for building ontologies is presented in [65]. Such ontologies contain large amounts of factual data, whereas the schema part is relatively small. Due to the employment of automatic extraction tools they do not require much maintenance effort. With the improvement of methods for automatically extracting semantic metadata, more resources can be successfully converted to ontologies. The proposed DBpedia conversion tool [17] opens a possibility to create an ontology out of any Wiki content and integrate it with other ontologies on the web. The tool allows creation of an ontology from the English version of Wikipedia [7], which is used as an example of the categorization ontology in the method of automatic text categorization described in this dissertation.

An ontology intended for text categorization should achieve a balance between a large number of instances and instance relationships, and comprehensive schema. A large instance base with labeled, highly connected named entities is essential to accurately transform analyzed documents into semantic graphs. The labels should include entity names, different alternative names, commonly used abbreviations and others. On the other hand, a well-developed schema with a proper taxonomy is responsible for the categorization quality. The precision in categorization of individual entities is essential for the classification algorithm. A well-developed schema with deep and wide taxonomy facilitates precise classification of entities to the most specific categories, and as a result, it allows the categorization algorithm to achieve better accuracy. The task of an ontology-based text categorization requires named entities to be recognized in the classified documents and thus the proposed method is limited to the domains that already have a well-developed ontology.

2.3. Ontology Storage and Query Languages

Ontology defines knowledge of a specific domain. For use in the computer applications, it requires proper storage, capable of maintaining an ontology of a suitable size, and supporting ontology access with a specialized query language. The requirements for a good ontology storage engine go beyond storing factual knowledge and meta-information. An ontology in RDF/S or OWL has a specific logic defined by the choice of the ontology language. A good ontology storage engine must support logic defined by RDF/S or the selected OWL flavor in order to allow correct interpretation of the stored domain knowledge.

In most cases, the current ontology storage engines rely on the already implemented relational databases with SQL access, as triples can naturally be converted into relational tables. The used storage strategies vary from a single table of triples to a complex combination of several relational tables that is driven by the ontology schema definition. Among the most commonly used storage systems are Jena [121], Sesame [26], KAON2 [118] and Redland [19]. They support both in-memory storage for smaller ontologies and persistent storage for very large ontologies (ontologies that can reach over hundreds of millions of triples). They tend to work efficiently for small to medium sized ontologies, when they use mainly in-memory structures. However, there is a significant performance degradation in case of intensive querying of large ontologies that require accessing persistent storage. Data has to be brought to main memory from persistent storage, and algorithms that perform large volume of queries may become orders of magnitude slower. All of the ontology storage engines come with a low-level API to access the resources, implemented logic for the selected parts of the ontological framework, and a query language built on top of it.

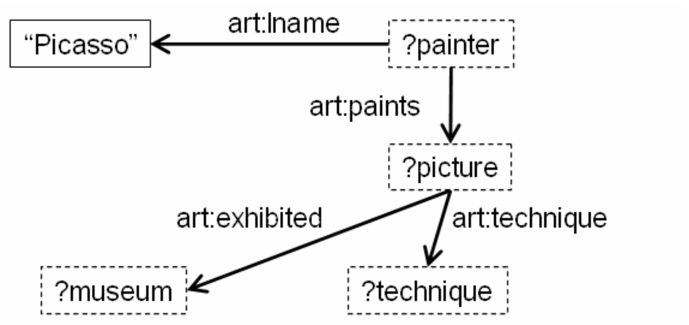
The logic layer is implemented on top of low-level storage or sometimes partially incorporated in it. Such a layer is needed to simplify ontology query languages and provide the required entailment. Integrating parts of the framework logic into the low-level storage in most cases comprise of statement materialization according to the ontology rules. It is not commonly used outside of the schema part of the ontology, which tends to be relatively small compared to the number of instances, and in many cases is restricted only to the taxonomical relationships. The ontology entailment is handled by specialized reasoners that implement selected flavors of the description logic, as it can be reduced to the satisfiability checking of a specific subset of description logic [50]. Several effective reasonsers have been created for RDF/S, OWL-Lite and

OWL-DL. There is no efficient reasoner that fully supports OWL-Full, as it is proved to be undecidable [50]. It can be modeled by the first-order logic [79], or the first-order logic with reified predicates [120]. However, most of the real-problems can be successfully solved using the expressive power of OWL-DL. The reasoners are used both to check the consistency of the ontology as well as the answer queries that include inferred statements, which are not originally present in the knowledgebase.

RDF/S is supported in Redland by the Rasqual query engine [20] and natively in Sesame. OWL support can be added to Sesame with use the of OWLIM implementation [61]. External OWL reasoners, such as Racer [39] and Pellet [104], can be added to Jena. Native OWL-DL reasoning is now supported in Jena and KAON2.

Ontology query languages enable us to retrieve and manipulate knowledge that is stored in an ontology. They focus on defining graph patterns and expressing various restrictions on entities and relationships participating in the defined patterns to extract the exact information the user is interested in. Query languages do not offer any support for the logic defined in the accepted ontology reasoning models. They all rely on the implementation included in the underlying storage systems and reasoners.

Currently, there are several ontology query languages available. Among the most important and commonly used there are SPARQL [84], RQL [60], RDQL [95] and SeRQL [25]. After years of development of a variety of query languages for ontologies, finally in the beginning of 2008, SPARQL became an official W3C recommendation. All ontology query languages offer capabilities for searching graph patterns with given restrictions. A graph pattern is a template of a graph structure represented as a collection of adjacent triple patterns, where both edges and vertices can be either variables or constants. They are used to define both structural and semantic properties of information to be searched. In Figure 2.3, we present an example of a SPARQL query and its result based on the ontology fragment depicted in Figure 2.1.



```
SELECT  ?museum, ?technique
WHERE { ?painter art:lname "Picasso" .
        ?painter art:paints ?picture .
        ?picture art:exhibited ?museum .
        ?picture art:technique ?technique }
```

```
museum = <.../www.museum.es/>
technique = "oil on canvas"
```

Figure 2.3 SPARQL graph pattern and query result

Apart from the major languages, there are several others offering very useful extensions and constructions. Especially interesting are the extensions related to semantic association discovery, as part of the ontology-based text categorization utilizes this feature. Some of the ontology query languages, such as G+ [29] and GraphLog [28] offer support for defining regular path queries. Partial support is included in SeRQL [25] or PSPARQL [10]. Unfortunately, none of them can fully support searching for semantic associations with specific restrictions imposed on properties and entities included in the path, thus later in this dissertation we present SARQLeR, our extension of the SPARQL language.

2.4. Classical text categorization methods

Text categorization is a task of assigning one or more predefined categories to the analyzed document, based on its content. The use of machine learning algorithms in the field of automatic text categorization is very common and it generally provides very good results. Majority of the existing algorithms for text categorization can be classified as supervised, unsupervised and semi-supervised methods. Unsupervised methods do not use any external information about a document and a corpora to perform classification. They try to find patterns hidden in the input data to group similar documents without assigning category labels. A classical example of the unsupervised text categorization is document clustering. Document clustering is a technique for unsupervised document organization, which discovers natural groupings of documents. Documents within each group are similar to each other, and dissimilar to documents in other groups. Clustering methods include hierarchical clustering [56], k-means clustering [44], and others. Supervised text categorization methods rely on labeled data to learn features of different categories and use them later to classify previously unseen documents. Classical supervised methods include probabilistic Bayesian models [68], nearest neighbors classifier [124], decision trees [73], and Support Vector Machines [57]. Semi-supervised methods combine the approaches to learn from labeled data and enhance the classification function with unlabeled examples. The idea of using unlabeled documents to improve categorization data is present in machine learning at least since 1968 [45]. For example, such data can be used in Transductive Support Vector Machines [58] and Naïve Bayes expectation maximization [76]. Comparative studies of different categorization methods are presented in [125], [27], [41] and many other publications.

Text categorization methods operate using different document features, thus they require a specific mapping of the analyzed unstructured text to the selected internal representation. A choice of mapping has to consider what meaningful textual units can be used as features by the classifier and what rules can be applied for their combination. Typical choices of features include words, phrases, and word senses. The most commonly used document mapping is a bag-of-words. Despite of the almost over-simplification, the mapping achieves very good results in text categorization. Other approaches to document mapping utilize more complex representations to capture the dependencies or relationships that are lost with the use of bag-of-words approach. Extensive tests performed by Moschitti and Basili [74] in 2004 showed that the use of complex features in classification does not necessarily produce significantly better results. On the other hand, more complex features such as relationships, phrase distance, and word co-occurrence are much more important in graph-based or semantically-enhanced approaches. One of the approaches to represent a document as a complex structure is a graph representation presented in [92]. The authors propose to treat words as nodes in the graph and connect them in the natural order, as they appear in sentences. This approach allows discovering of phrases by indicating frequent paths in the resulting directed graph and adding them later as features for categorization.

Text categorization, by the nature of the problem, has to deal with high-dimensional feature space in the document representation. Each choice of document representation has to be accompanied by an approach to dimensionality reduction in order to lower the processing complexity, prevent overfitting, and concentrate on the most important features of the document. The most common method is to use the document and term frequency to estimate feature importance. Other methods include information gain, clustering of similar terms, Latent Semantic Analysis [31], and methods based on link analysis for graph-based algorithms.

A comprehensive survey of text categorization methods, used document representations and feature selection is presented in [96].

The extraction and use of document features, such as word distance or common phrases, can be achieved using only the information contained in the document corpora. Subsequent steps of incorporating linguistic or semantic features into text categorization require the use of some kind of background knowledge, such as ontology or statistics from selected language corpora.

2.5. External Knowledge in Text Categorization

External or background knowledge can improve text categorization, especially for short or ambiguous documents [127] [23]. It helps to unify the vocabulary, match important phrases, strengthen co-occurrences, or use related information not included in the original document in order to perform document categorization.

WordNet [8] is one of the best known sources of external knowledge used for text categorization. It is a network of semantically related words, groups of words, and phrases. WordNet was originally created as a combination of a dictionary and a thesaurus, but it expanded far beyond the initial scope. Words are grouped into semantic groups (*synsets*) that provide word synonyms together with short explanations and general definitions. Nouns and verbs are organized into hierarchical structure (*hypernyms*). These features can be used to find and match similar words and to unify and reduce the vocabulary during the text categorization process. Additional lexical relationships such as meronym, troponym, verb entailment, and others represent complex dependencies between the described elements. WordNet, having well-defined relationships among words, phrases and group of words, can be used as a high-quality and comprehensive lexical ontology of the English language. For the text categorization purposes, it is used mostly to unify the vocabulary across the documents by modifying the document features with use of the related words. WordNet is successfully used in document clustering presented in [51] to leverage the semantics of words and in the categorization algorithm described in [23]. External knowledge may be used to collapse some terms into a common class or a term to strengthen the selected features. For example the terms 'pork' and 'beef' are similar, as they are both defined as sub concepts of the word 'meat' in WordNet. Another approach of utilizing WordNet in text classification is described in [89], where *synsets* are used to disambiguate sense for different terms in the analyzed document.

A natural next step beyond the related words is the utilization of knowledge in the classification task to find associated entities and concepts. Ontologies offer knowledge of a given domain that is organized both in a structural and semantic way. Their use in text categorization and topic identification has lately become an intensive research topic. As ontologies provide named entities and relationships between them, an intermediate categorization step requires matching terms to ontological entities. Afterwards, an ontology can be successfully used for term disambiguation and vocabulary unification, as presented in [23]. The authors propose the use of synonymous words, multi-word phrases, generalization and polysemous words to overcome deficiencies present in systems where features are only word-based. Another approach, presented in [75], reinforces the co-occurrence of certain pairs of words or entities in the term vector that are related in the ontology, but are not clearly related in the document corpora.

Further utilization of the ontological knowledge is presented in [101] and [40]. The ontology not only contains named entities and helps to recognize them, but more importantly, it describes relationships among them. It can help to discover multi-word entities in the document, which otherwise would be treated as separate words. Later, even properly disambiguated named entities can be assigned to multiple contexts. Recognizing a proper context with the help of ontological entity classification and other entities present in the document can efficiently narrow down the target categorization domain and improve accuracy. Text categorization with the help of a dynamically obtained ontology is described in [123]. The extraction of text features and keywords from the training corpus is used to create a map of associated terms. They are transformed later to ontology and association rules to enhance the standard categorization process.

Text categorization deals with natural language texts that contain terms from a general or specialized knowledge. For humans, an encyclopedia is one of the important knowledge sources for finding a proper definition or interpretation of a given, unknown name or entity. Automatic text categorization can also benefit from using encyclopedic knowledge that is encoded in the form of an ontology. Such an ontology should define names, terms, categories and relationships from general human knowledge. The largest undertaken projects to create such a knowledgebase include Cyc [67], WordNet, and Wikipedia.

Wikipedia. Initial work has been done lately on using Wikipedia as an ontology for the categorization purposes. Conversion of Wikipedia from a structured XML document into an RDF/S ontology is possible due to the work initiated by Auer and Lehmann, described in [17], whereas the final goal is to produce a semantic Wikipedia, as described in [116]. Text categorization approaches utilize the fact that Wikipedia contains a vast amount of general knowledge that is interconnected and categorized. It spans over multiple domains, many of which are well covered and contain very detailed descriptions. Entries, represented as individual Web pages in Wikipedia, can be treated as named entities and the categories form a kind of thesaurus that is a mixture of a taxonomy and collaborative tagging [119]. Although the category graph cannot be directly transformed into a taxonomy, the work presented in [82] shows some solutions to overcome this issue. The authors also describe a method for creating additional taxonomic relations between instance entities, directly from the entity descriptions. The analysis presented in [128] shows that Wikipedia resources can be successfully used for various NLP and text categorization tasks. Semantic relatedness, presented in [106], can replace WordNet in classification and even outperform it. Wikipedia can be also used as a source of knowledge about the semantically related neighbor entities. The authors in [32] propose to use it to enhance the categorization with available description of *similar neighbor* entities. The use of directly related entities helps to tackle one of the most important problems in natural language processing – *polysemy.* With proper selection of entities and Wikipedia categories, it is possible to disambiguate the meaning of a selected entity and to put it into a proper context.

For the scope of this dissertation, successful use of ontological features and the encyclopedic knowledge in text categorization is an important indication that ontology-based categorization may be possible and is worth further investigation. It shows that the incorporation of well-defined external knowledge can be used in text categorization to improve the classification quality [48]. Successful use of Wikipedia in text categorization tasks, as presented in [32], makes it a promising ontology for the forthcoming experiments.

2.6. Natural Language Processing (NLP)

Natural language processing (NLP) lies on the intersection of computational linguistics and Artificial Intelligence. Understanding and processing of natural language is one of the important and also very hard problems in Computer Science. Sometimes, it is referred to as an AI-complete problem as a parallel of NP-complete problems, a well-known class of very difficult problems from a computational point of view. Assigning natural language understanding to group of AI-complete problems suggests that solving it is as hard as making computers as intelligent as people. Among other different major tasks of NLP, information retrieval, information extraction, and named entity recognition and disambiguation are directly relevant to the ontology-based categorization.

Natural language parsing and even partial understanding requires at a minimum information about the language grammar, syntax, word semantics, and the available vocabulary. The work on parsing of natural languages started in the 1950s and still is the field of intensive research, yet the understanding of natural language is still not possible. One of the basic building blocks for all NLP applications is sentence parsing. In general, sentence parsing is a two step process that consists of part-of-speech tagging and building a parse tree. Part-of-speech (POS) tagging is the first step in parsing natural languages. It is a process of annotating words in sentences with appropriate parts of speech, based on the word definition and the surrounding contextual information. For this purpose, annotated corpora have been developed for different languages. Brown Corpus [64] was the first corpora for the English language and was successfully used for many years. Nowadays, we can choose from the International Corpus of English [33], the BYU Corpus of American English [2] and the largest one, British National Corpus [1], published in 2005. POS-taggers use different approaches to assign and disambiguate parts of speech, such as corpus statistics, hidden Markov models, dynamic programming, encoded rules and others. Among the popular POS-taggers are Stanford [111], CLAWS [66], TreeTagger [93], and Tsuruoka’s part-of-speech tagger for English [112], used for the prototype implementation in this dissertation.

Sentence parse trees present the syntactic dependencies between the previously tagged words in sentences. In the majority of cases, parsers and taggers use information obtained by analysis of one selected language corpus. Majority of the currently available parsers are either based on the statistical information from the corpora, or combine such information with other approaches. This includes the use of neural networks, probabilistic context-free grammars, dependency grammars or link grammars. Our prototype implementation uses s chunk parser created by Tsuruoka [113] that is based on the analysis of sentence chunks of different length with maximum entropy classifiers to recognize phrases.

The information from a syntactic sentence parse has been successfully used for the extraction of terms and relationships. Examples from the field of Bioinformatics show that with the help of a proper dictionary of terms and predefined relationship names, software systems are able to identify semantically significant associations between entities. In [110], the authors used a context free grammar with the dictionary of terms and synonyms to discover protein interactions from unstructured text. Some newer approaches leverage the knowledge from ontology to improve the process of entity recognition and association discovery. In [85] and [88], the authors utilize information from UMLS (Unified Medical Language System) [78] and MeSH (Medical Subject Headings) [77] to extract even complex named entities and discover relationships between them. These approaches take advantage of the domain specificity that largely uses terms from a controlled vocabulary. The use of such a vocabulary makes the problem of recognizing and disambiguating named entities easier. Even compound entities that are defined as a structure of a few simple entities, such as *adenomatous hyperplasia of the endometrium*, can be successfully recognized. Relationships are extracted with the help of syntactic parsing information and list of known associations, after the named entities have been properly recognized and marked in text.

In one of the phases of the ontology-based text categorization presented in this dissertation, we focus on finding associations between named entities in the analyzed document. Our approach follows the idea for relationship extraction using previously recognized named entities and information from the parse tree.

CHAPTER 3 EFFICIENT RDF/S STORAGE: BRAHMS

In this chapter, we present BRAHMS [52], a main-memory efficient storage system for RDF/S. BRAHMS was designed as a high-performance RDF/S storage system capable of handling semantic association discovery in very large ontology. High-performance and memory-efficiency were the priorities in designing BRAHMS system, and a specific design and optimization decisions have been taken to shape the system to its best performance. In the design process, we concentrated on maximizing the performance of the basic API calls for searching and retrieving ontology statements, as they are the building blocks for more complex algorithms. Memory efficiency was also a priority, as minimizing the memory footprint enables working with larger ontologies and leaves more available memory for executing user algorithms. The use of a high-performance and memory-efficient RDF/S storage such as BRAHMS, not only allows for fast execution of association discovery algorithms, but also enables to run them on large ontologies and discover long associations in a reasonable time. In this chapter, together with the motivation need for such system we present details of the design and implementation of BRAHMS. Performance tests and evaluations are described at the end of the chapter.

3.1. Semantic association discovery

Ontology provides a machine understandable description of a domain on two levels: conceptual and factual. The ontology schema enables proper interpretation of factual knowledge in the description base. Knowledge in the ontology is represented on both levels as statements, each specifying a named relationship between uniquely identified resources. A statement, alternatively named a *triple*, consists of a subject, a predicate and an object. The subject and object are resources or literals in the ontology. The predicate defines not only relationship between these two elements, but also specifies the directionality of this association. An ontology that is a set of statements consisting of resources connected to each other by directional relationships, can be naturally interpreted as a labeled directed graph.

A semantic association path connecting two entities, as defined in [16], is a sequence of meaningful relationships connecting the two entities. Searching for semantic associations is one of the key elements of the knowledge discovery in the ontology. A semantic association describes how the two entities relate to each other, what intermediate entities are involved and by interpretation of the included properties, what are their roles.

One of the fine examples of a meaningful semantic association is a dependency discovered between *magnesium* and *migraine* by Dr Don R. Swanson [108] while he was focusing on the literature-based discovery in the biomedical domain. There was no publication that directly related migraine to low level of magnesium or even mentioned both terms together. After manual analysis of many publications in PubMed [5], he presented a collection of research results from different publications, where each document contained an important fragment of the association. Arranging them in sequence allowed creating a meaningful semantic association between migraine and magnesium, to support the hypothesis that receiving larger doses of *magnesium citrate* may alleviate migraines. If we can extract the same knowledge from publications in the biological domain and transform it into an ontology, as proposed in [85], this semantic association can be represented as a path of relationships and entities connecting magnesium and migraine.

Let us define formally the semantic association using the graph terminology. We will assume that R is an RDF description base and $r\ p\ s$ is a triple in R, where r is a subject resource, p is a named relationship directed from r to s, and s is an object resource.

Def. 3.1 A directed path between resources r_0 and r_n in R is a sequence $r_0\, p_1\, r_1\ \ p_2\, r_2\, ,\ \ldots\ \ p_{n-1}\, r_{n-1}\, p_n\, r_n$ (n>0) if $r_0\, p_1\, r_1,\ \ r_1\, p_2\, r_2\, ,\ \ldots\ \ r_{n-2}\, p_{n-1}\, r_{n-1},\ r_{n-1}\, p_n\, r_n$ (n>0) are triples in R. The length of the path is n. Moreover, we require that all of the resources r_i ($0 \leq i \leq n$) in the path be distinct (we will only consider simple paths). □

Def. 3.2 An undirected path between the resources r_0 and r_n in R is a sequence $r_0\, p_1\, r_1\, p_2\, r_2\, ,\ \ldots\ \ p_{n-1}\, r_{n-1}\, p_n\, r_n$ (n>0) if for each property p_i and the two neighboring resources $r_{i-1}\, p_i\, r_i$ ($0 < i \leq n$) in the path, either $r_{i-1}\, p_i\, r_i$. or $r_i\, p_i\, r_{i-1}$. is a triple in R. We will consider only simple undirected paths. □

Def. 3.3 Two resources r and s in R are semantically associated, if there exists an undirected path in R connecting the two resources. □

Discovery of semantically related entities and interpreting the meaning of located associations is a frequent task in Risk Assessment, Anti-money Laundering [97], or Threat Assessment [100]. Usually, we search for semantic associations of unknown or variable length. The included properties can be of any directionality, as the two resources may not be linked by a directed path, but they may be connected by a path that includes inverse relations, representing potentially vital information about the semantic linkage existing between the resources in question. In majority of practical applications, the discovered association paths should be built from instance resources, as they represent facts in the knowledge base. Including schema classes in the association path leads to mixing factual knowledge with meta-information and may produce less meaningful associations. Including literal entities as intermediate nodes in the association path may be important in some cases (e.g. two resources have the same name or a specific value), but in general, relationships defined in the ontology between entities are the main point of interest. Still, although literal values and schema types should not be used as the building blocks of the path itself, they represent important and valuable information for understanding the meaning of the associations.

3.1.1. Query languages for semantic association discovery

The problem of semantic association discovery cannot be solved with the available high-level ontology query languages, as they are not designed for such purpose and lack the necessary capabilities [14]. All of the RDF base query languages allow the specification of certain fixed patterns of semantic associations between entities, as well as expressing various restrictions on the relationships and intermediate entities participating in associations. However, a semantic association can be of unknown length and include different unspecified properties and connecting entities. A semantic association can be treated as an undirected path, so the directionality of the properties cannot be determined a priori. The main problem is that the created path expressions in the ontology query language can match only paths of a fixed length and of specified directionality of the participating relations. A flexible pattern of specifying semantic associations cannot be expressed using the currently supported fixed-patterns of RDF query languages. Let us demonstrate it on the example of finding all paths of length up to two between two resources A and B in a very simple graph.

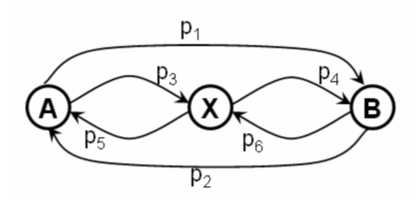


Figure 3.1 Sample graph

The semantic associations located in the graph shown in Figure 3.1 (all up to length two) are presented in Figure 3.2.

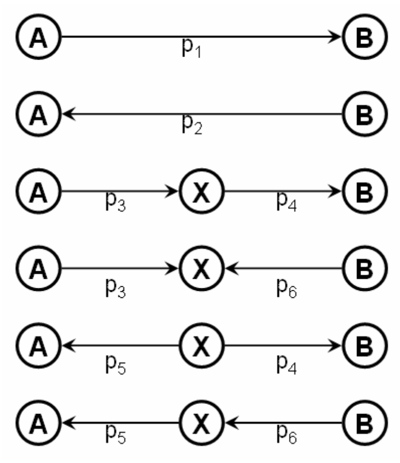


Figure 3.2 Located semantic associations

They can be seen as templates of semantic associations up to length two. Each of them can be easily expressed in an ontology query language. In SPARQL, they can be written as presented in Figure 3.3.

```
SELECT ?property_1
WHERE { A ?property_1 B }

SELECT ?property_1
WHERE { B ?property_1 A }

SELECT ?property_1, ?x, ?property_2
WHERE { A ?property_1 ?x .
        ?x ?property_2 B .
        FILTER ( A != ?x && B != ?x ) }

SELECT ?property_1, ?x, ?property_2
WHERE { A ?property_1 ?x .
        B ?property_2 ?x .
        FILTER ( A != ?x && B != ?x ) }

SELECT ?property_1, ?x, ?property_2
WHERE { ?x ?property_1 A .
        ?x ?property_2 B .
        FILTER ( A != ?x && B != ?x ) }

SELECT ?property_1, ?x, ?property_2
WHERE { ?x ?property_1 A .
        B ?property_2 ?x .
        FILTER ( A != ?x && B != ?x ) }
```

Figure 3.3 SPARQL queries for semantic associations up to length 2

As shown above, to express the search for semantic association up to length two, six different queries are required in SPARQL. As the path length increases, the number of the required queries grows exponentially, due to the non-directionality of relationships. Additionally, because we search for simple paths, each query must have conditions which guarantee that each resource appears only once in a given path. As a result, even though it is possible to discover semantic associations only up to a fixed length using the current RDF query languages, it is prohibitively expensive.

3.1.2. Ontology storages and semantic association discovery

As discussed in the previous section, higher-level ontology query languages are incapable of semantic association discovery. One possible solution is to create graph-based algorithms that utilize API-level graph primitives. An appropriate storage should offer a lower-level API to operate directly on graph structures to search for node neighborhood in the ontology. In a very basic implementation, the semantic association discovery algorithm should be based on graph traversal methods, such as the depth-first search or breadth-first search. Implementations providing such a low-level API include Jena [70], Sesame [26] and Redland [19]. The three systems are among the most widely used nowadays.

Given the complexity of finding simple paths in a graph, the only solution for a reasonable implementation of the semantic association discovery algorithm is a very fast querying mechanism for given triple patterns (subject, predicate, object). Achieving a high speed in this basic algorithm is the key for making semantic association discovery usable, as longer associations in large graphs may require hundreds of millions of possible paths to check. The optimum solution requires a system to keep the whole graph structure in main memory, since accessing the disk-based or remote databases significantly slows the search process. All of the storages mentioned above offer both in-memory and database storage for ontologies. Unfortunately, all of them have certain drawbacks and limitations when it comes to discovering longer semantic associations in large ontologies.

In-memory storage implementations in the presented systems are designed to handle small ontologies, mainly for test purposes. They either lack proper indexes or the used data structures require massive amounts of memory. In case of small ontologies, the memory footprint is acceptable, whereas large ontologies cannot fit into the physical memory of typical computers. The only solution is in using a database implementation which can handle them, but at the price of significantly lower performance.

The lack of an ontology storage capable of performing semantic association discovery in a reasonable time in larger graphs created the need for a specialized, high-performance RDF/S storage. Our work resulted in creating a system that is capable of handling large graphs using in-memory structures which is optimized for read-only querying.

3.2. BRAHMS design and implementation

Lessons learned from working on the semantic discovery project [98] helped us define the critical features of an efficient ontology storage, suitable for semantic association discovery:

- efficient in-memory data structures for graph representation to minimize the memory footprint,
- highly optimized indexes for querying for basic patterns in triples,
- system optimized for read-only querying a fixed ontology, not for modifying it.

BRAHMS has been designed to be a fast main memory-based storage system for RDF, capable of storing large description bases and serving as a base for efficient implementation of semantic association discovery algorithms. We decided to support only RDF and RDFS ontologies, as reasoning associated with OWL restrictions and class definitions would significantly complicate the storage design and structures, inevitably degrading the performance. The choice of using only the main memory for storing graph information was dictated by the requirement of maximizing access speed. To avoid slower access to disk or a database, all ontology structures have to fit into main memory. Ac a consequence, the achieved speed comes with the price of limiting the size of an ontology that can be used.

The memory usage restriction required us to use a compact representation of the triples, nodes, their values, and storing only the basic indexes for accessing data. As the system was designed for read-only ontology querying, we were able to forgo dynamic indexes and data structures in favor of the static ones. All ontology information and indexes have to be calculated and compacted beforehand to minimize memory usage during the query phase.

For large ontologies, the process of loading and calculating the required indexes is quite expensive. To avoid a lengthy system startup time, we decided to make the in-memory representation of the description base read-only and rely on the pre-built memory image of the ontology storage for querying. Such an approach allows us to optimize the memory usage by utilizing specialized data structures and also to create all of the indexes only once. In the proposed solution, updating ontology is not possible during the search process. Each time the ontology changes, the image has to be re-created and loaded into BRAHMS.

BRAHMS uses the Raptor RDF parser [87] for the initial load of the RDF file into the internal memory structures. C++ was our choice for implementation language due to the offered speed and direct, low level memory management. The construction of the memory graph representation that can be successfully reused after being written to an image file on disk requires an internal representation which does not rely on direct memory references, as they are volatile. Instead of direct memory pointers to resources and indexes, we used a continuous virtual memory block with only one memory reference and represent all internal references as appropriate offsets. This allowed for a compact in-memory representation, while not using volatile pointers and not sacrificing the access speed to resources and indexes.

Fast and efficient indexes for accessing specific statements in a description base are the key for creating efficient semantic association discovery algorithms. Each resource in the ontology can be uniquely identified by its URI, which is a string. One of the most common operations in the planned search algorithms was comparing resources to check if they are identical. String comparison is very slow in contrast to comparisons of primitive types, especially that in most cases in the ontology, resource URIs share a common prefix. To avoid costly comparisons, we decided to uniquely identify each resource using an integer value. As a result, a simple comparisons and resource sorting can be performed much faster. To provide an efficient mapping between resource URIs and integer identifiers, we added hash tables for matching string URIs with resource nodes. The structure of internal tables in BRAHMS for statements, resources, indexes and string values is presented in Figure 3.4.

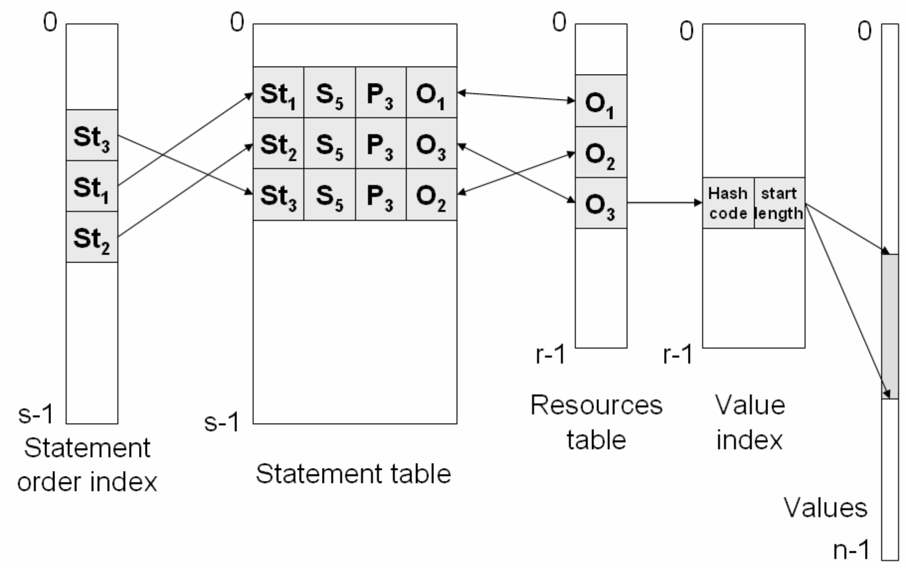
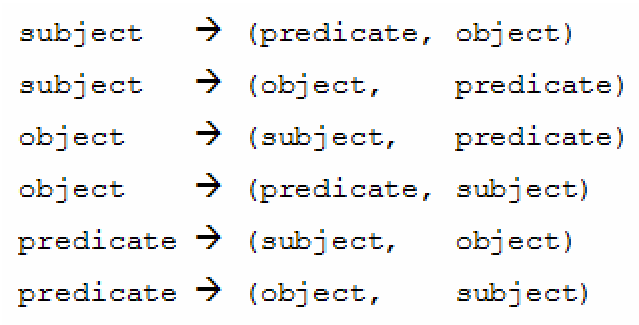


Figure 3.4 Internal structure of tables in BRAHMS

Each resource, class, property and literal is identified by a unique numeric identifier. To minimize the memory usage, the internal data stores operate on these identifiers, keeping their string values in separate tables. BRAHMS uses the following types of internal data stores:

- the list of triples that contain only numerical identification of resources, properties, classes or literals together with indexes for fast access to them,
- the list of resources, properties, classes and literals that match ID with proper label/URI, and
- the list of resource values (URIs) and literal values.

Taking into consideration the types of algorithms that would be implemented using BRAHMS and the memory size limitations, we have created basic node-centric indexes that can be used to implement more complex queries and algorithms. The statements (triples) are accessible using the following indexes:



```
subject   → (predicate, object)
subject   → (object,    predicate)
object    → (subject,   predicate)
object    → (predicate, subject)
predicate → (subject,   object)
predicate → (object,    subject)
```

Figure 3.5 Simple statement indexes in BRHAMS

It means that knowing the subject of a statement, we can get all statements with this subject. Additionally, the returned set of statements can be ordered by predicate and than by object (subject → predicate, object) or by object and than by predicate (subject → object, predicate).

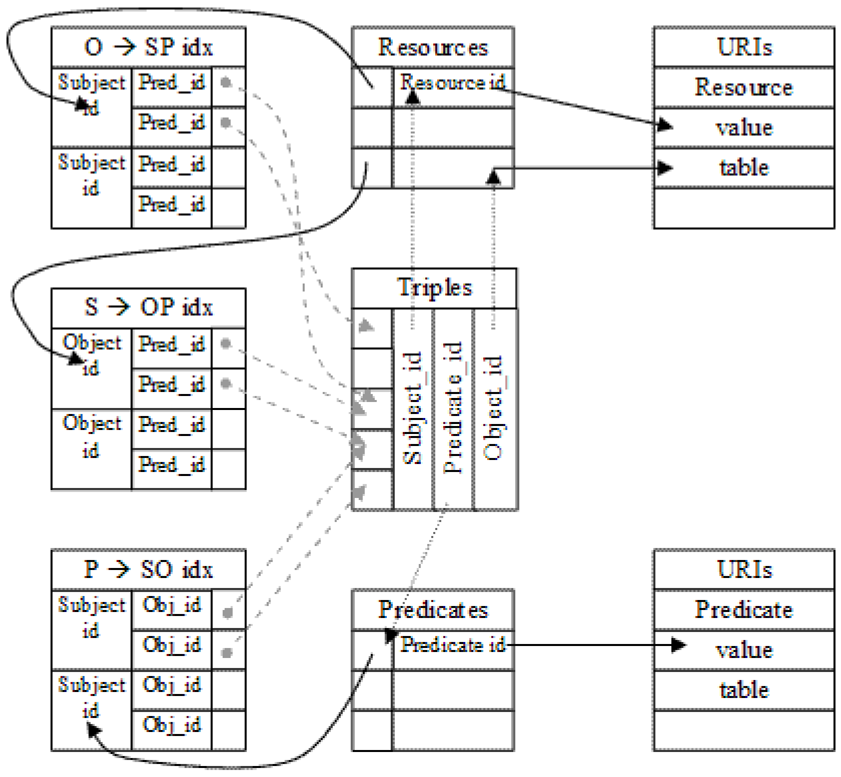


Figure 3.6 Indexes and connections in BRAHMS

These six indexes cover all possibilities of indexing triples and are used for a fast retrieval of node neighborhoods, as well as for searching for and merging of neighborhoods during the semantic association discovery process. Given the identifier of a resource, the first matching statement is accessed in constant time and the following statements are already ordered in the index table. Compound indexes are easily derived based on the simple ones:

```
(subject,   object)    → predicate
(subject,   predicate) → object
(predicate, object)    → subject
```

Figure 3.7 Compound statement indexes in BRAHMS

It means that knowing a combination of any two elements of the statement (subject, predicate, object), we can get all of the statements that match this pattern. They utilize the sorting order of primary indexes.

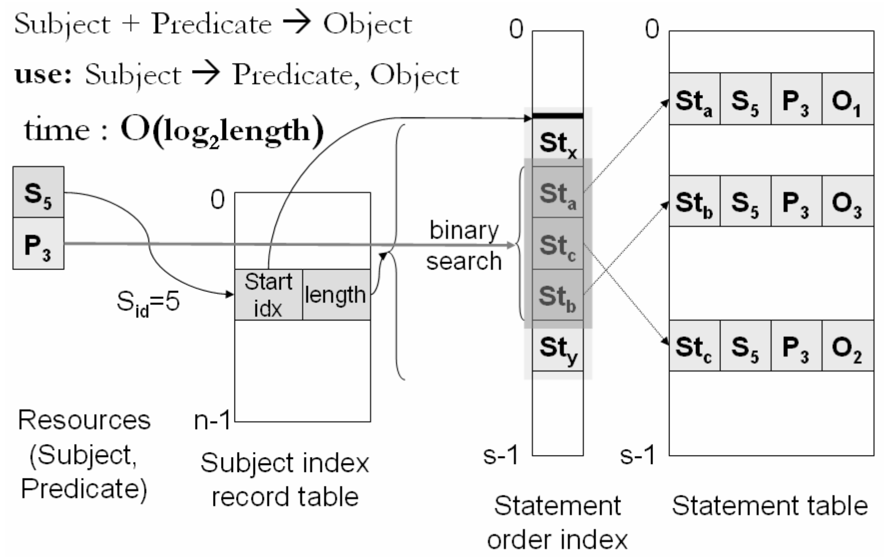


Figure 3.8 Compound index access pattern

Access time for compound indexes is bound by the binary search for the upper and lower border on the secondary element in index. The rest of the access pattern remains the same.

Another design decision was to separate the instance resources, properties, literals, and classes as they represent different pieces of information. In majority of cases, semantic association discovery operates on the instance description base and does not use the schema classes or literals. This split was created in order to optimize the search algorithms to remove comparisons among resources of different types. Still, statements with schema classes or literals can be accessed using separate operations, but they are not mixed with statements including only instances. As a result, literals, properties and schema type resources are kept in separate memory structures with their own, but similar indexes.

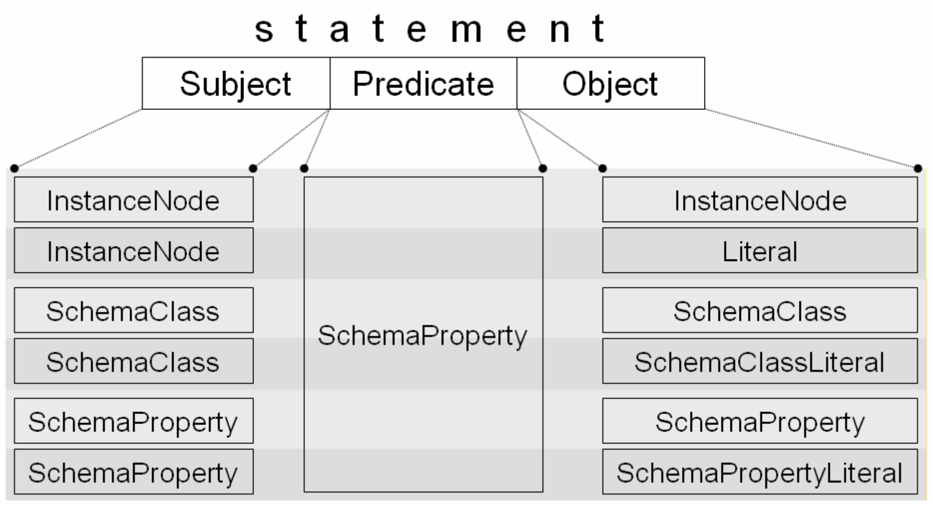


Figure 3.9 Statement and resource types in BRAHMS

Separation of different resource types resulted in creating several disjoint statement types to optimize access. Figure 3.9 presents the allowed combination of resource types that constitute statements in BRAHMS. Similarly to resources, each statement type has its own, separate indexes. Multiple statement types are the price for optimizing access time and removing unnecessary resources or statements from the most common operations.

Separate optimization was performed for handling the schema taxonomy of classes and properties. As the different types of resources have been separated, naturally the instance base and the schema represent different subgraphs. The only allowed connection between them is the `rdf:type` property. In an RDF/S file, only the direct types of an entity are provided, and class subsumption must be handled by the ontology storage. In BRAHMS, an instance "knows" its direct types (multiple types are allowed), and a class has assigned a list of its direct instances. The requests for types of a specific resource, or for getting instances for a specific class are among the most common. With the defined RDFS logic, a user expects that subsumption is used when handling such requests. To optimize such queries, a Kleene closure is calculated on the schema classes. Each class has a sorted list of its all ancestors and descendants, as well as its parents and children.

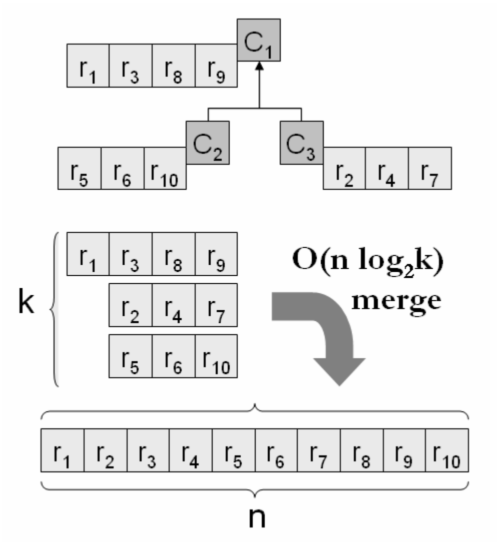


Figure 3.10 Aggregation of instances in class hierarchy

Answering a question if an instance is of a given type, results in a binary search for this class in the list of ancestors of the instance's direct type. The opposite queries, from a class to get all its instances, require merging the sorted list of associated instances of the class descendants, which is performed in O(n*log*k) time.

The presented design choices and the optimized implementation allowed creating high-performance main-memory storage for RDF/S ontologies. With BRAHMS, searching for longer semantic associations, even in bigger graphs, may be done in almost real time as shown later in Section 3.3 on performance evaluation.

3.2.1. C++ and Java API

BRAHMS has been implemented in C++. It offers a rich API to access resources and statements in variety of ways. The API provides all of the necessary primitives that allow constructing graph-based algorithms for search, traversal, and other purposes. A native API was used to implement the basic semantic association discovery algorithms, the implementation of SPARQLeR query language, and for the ontology-based categorization. The C++ implementation of BRAHMS is designed to work both in Unix and Windows environments.

Due to the high popularity of Java and the need for BRAHMS functionality in other developed projects, an overlay Java API has been designed and implemented. It offers the same functionality and access to the ontology resources as the native C++ interface. BRAHMS Java implementation supports SemDis API ver. 0.3 [6]. In contrast to C++, a user can develop a prototype solution faster, as all memory management and garbage collection has been taken care of by the Java language. Unfortunately, Java API cannot offer the same speed and memory efficiency as the native C++ API. It is caused by the necessary mapping of the resources in C++ to specific object instances in Java. Some of the speed is lost in the mapping, while the necessary Java objects require the additional memory. Despite the sub-optimal performance in Java, BRAHMS has been already successfully used in the Insider Threat project [11], which proved its value as an RDF storage system, offering the necessary foundation for the implementation of fast graph algorithms.

The BRAHMS library together with the Java API is distributed as open-source and is available for download from http://lsdis.cs.uga.edu/projects/semdis/brahms/

3.3. Experiments

In our tests, we have compared our own system, BRAHMS, with the three RDF storage systems discussed previously: Jena 2.1 (Java), Sesame 1.1 (Java) and Redland 1.0.0 (C). Our efficiency tests of the memory-based implementation of the ontology storage included the initial loading time of the ontology and memory consumption for a given ontology size. The performance tests used optimized implementations of the same semantic association discovery algorithms developed using different storages. All tests were performed on the main-memory implementation of the compared ontology storages. We used several ontologies of different sizes and distribution of resource connectivity to cover a variety of data sets. All of the tests were performed on a dual-processor computer system with the following configuration:

- 2 Intel(R) Xeon(TM) CPUs running at 3.06GHz; 4Gb memory (3Gb available for a single user process); 220GB of hard disk available
- Red Hat 9.0 Enterprise Linux operating system,
- Java SDK 1.4.1_02; 1800Mb of maximum heap size for loading the bigger data sets and 512Mb of maximum heap size for loading the smaller data sets,
- gcc (GCC) 3.2.2 20030222 (Red Hat Linux 3.2.2-5), C/C++ code compiled with the '–O6' optimization flag.

3.3.1. Test ontologies and resources

In the tests, we used both synthetic and real-life datasets. For real-life ontology, we used two different subsets of SWETO (Semantic Web Testbed Ontology) [12]. SWETO contains real-world information about publications in Computer Science, including titles, authors and co-authors, and other publication details. Synthetic datasets included ontologies generated using the Test Onotlogy Generation Tool (TOntoGen) [81] and Lehigh University Benchmark [38]. We have used the following datasets in our tests:

- "SWETO_small", a dataset of 14Mb, containing 187,507 statements, 55,876 unique resources with the average node degree of 2.16 in the biggest connected component,

- "SWETO_big", a dataset of 255Mb, containing 3,196,692 statements, 813,479 unique resources and the average node degree of 3.90 in the biggest connected component,
- "Bus-Sport-Ent", a synthetic dataset, generated using TOntoGen to include multiply classified instances from three schemas (business, sports and entertainment); 14Mb in size, containing 104,891 statements, 29,825 unique resources and the average node degree of 3.86,
- Univ(10, 0), a synthetic dataset from Lehigh University Benchmark, 105Mb in size, containing 630,753 statements, 207,427 unique resources and the average node degree of 6.08,
- Univ(50, 0), a synthetic dataset from Lehigh University Benchmark, 556Mb in size, containing 6,888,642 statements, 1,082,818 unique resources and the average node degree of 6.09,
- Univ(700, 0), a synthetic dataset from Lehigh University Benchmark, 7.8Gb in size, containing 46,265,985 statements, 15,170,940 unique resources and the average node degree of 6.10.

The details of the test datasets are presented in Figure 3.11.

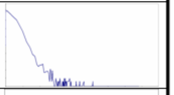
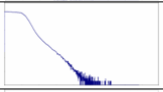
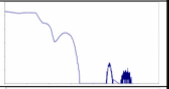
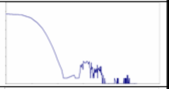
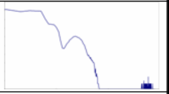
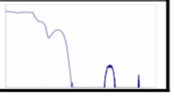
Dataset name	Instance Statements	Instance Nodes	Avg Node Degree	RDF File Size	Degree distribution log / log scale
SWETO medium	59,105	55,876	2.116	14 Mb	
SWETO big	1,553,112	813,479	3.818	255 Mb	
Univ (50, 0) *Lehigh University*	3,298,813	1,082,818	6.093	556 Mb	
Bus-Sports-Ent *TOntoGen*	45,000	29,889	3.011	13 Mb	
Univ (10, 0) *Lehigh University*	630,753	207,427	6.082	105 Mb	
Univ (700, 0) *Lehigh University*	46,265,985	15,170,940	6.099	7.8 Gb	

Figure 3.11 BRAHMS test datasets details

The largest dataset, Univ(700,0), was used only with BRAHMS to test the capabilities of the system. The other tested ontology storages could not successfully load it into their main-memory graph implementation.

Dataset	Starting resource	Ending resource
SWETO (small and big)	http://lsdis.cs.uga.edu/proj/semdis/testbed/#SWEET_215003 Chee-Keng Yap	http://lsdis.cs.uga.edu/proj/semdis/testbed/#SWEET_949653 Ravi Ramamoorthi
Synthetic TOntoGen	http://lsdis.cs.uga.edu/semdis/sports/Athlete_7271	http://lsdis.cs.uga.edu/semdis/business/Spokesperson_7611
Univ(x, 0) Lehigh University Benchmark	http://www.Department0.University0.edu/FullProfessor0	http://www.Department0.University9.edu/FullProfessor0

Figure 3.12 Resources used for semantic association discovery tests

In the semantic association discovery tests, we were searching for simple undirected paths between two selected instances in the ontology. The resources used as endpoints of paths in the selected ontologies are presented in Figure 3.12. For both SWETO datasets, we used the same endpoints, as presented in the on-line demo of the semantic association ranking. The dataset and endpoints for the synthetic TOntoGen ontology were taken from [86]. The endpoints for Lehigh University Benchmark synthetic datasets represent two professors from distant universities. We ensured that the semantic associations between the selected entities exist and the minimum distance between the selected entities is not too low.

3.3.2. Semantic association discovery algorithms

The implementations of the semantic association discovery algorithms were based on the depth-first search and bi-directional breadth-first search. Both are exhaustive algorithms that find all simple paths between two entities in the graph. These algorithms serve as the main building blocks for many more complex graph exploratory algorithms. The timing results of their runs should provide a good indication of possible performance of other graph algorithms implemented using the BRAHMS storage. For practical reasons, they impose a limit on the path length.

The depth-first search version uses a very limited amount of memory to hold only a stack of elements in the path, but it comes with the price of significantly slower performance. The bi-directional breadth-first search looks for the semantic association paths by growing frontiers from both endpoints.

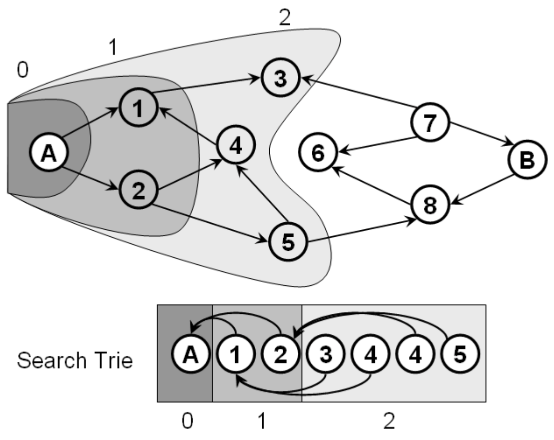


Figure 3.13 Frontiers of breath-first search in the graph and associated search trie

At each step of the bi-directional breadth-first search, a frontier of instances is expanded to the nearest neighbor nodes of the current frontier. The frontier is grown interchangeably from both path endpoints. Only the nodes that do not occur on the current part fragment are added to the next frontier. All information from the previous steps is kept in a *trie*, where it is possible to reconstruct a partial path from the frontier node to the initial search point. A newly created frontier is sorted and a join of two frontiers is performed to find candidates for semantic associations. For each located intersection point, a final check is performed to verify if the found path is a simple path. Storing the information about multiple path fragments at the same time requires a significant amount of memory, but results in a much faster execution time than the depth-first search version.

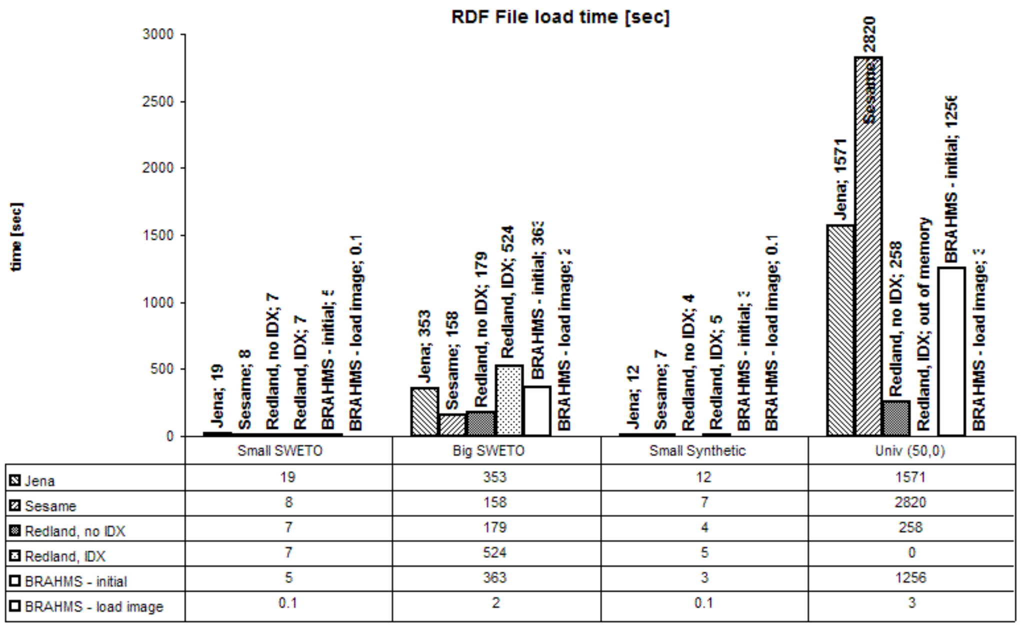
3.3.3. Ontology load experiments

Our initial tests included loading an RDF/S ontology into storage using the main-memory implementation, in order to estimate the memory consumption and the required load time. The process included the parsing of the RDF file and creating the memory structures. This represented the time needed for a cold start for each of the systems. The amount of memory required to represent each of the datasets as the internal structures of the storage represents the level of compactness of the used data structures. It may also determine if the memory exhaustive semantic association discovery algorithm would have enough free space for the expansion.

The comparison of the load times and the memory footprint for the tested storages is presented below. For BRAHMS, we included the time needed for creating a memory snapshot (cold-start) and using the previously created snapshot (warm-start). Similarly, two different values are also included for the Redland storage. Its main-memory implementation is designed for handling small ontologies and does not include proper indexes for statements. To make it comparable with other storages, we modified the implementation according to directions in [18] by adding the appropriate indexes.

We use the following labels in the presented experiment results:

- “Jena” – main-memory Jena implementation,
- “Sesame” – main-memory Sesame implementation,
- “Redland no IDX” – unmodified main-memory Redland implementation,
- “Redland IDX” or “Redland” – patched Redland with two additional indexes: subject to predicate-object, and object to predicate-subject,
- “BRAHMS initial” – cold-start of BRAHMS: parse and load the RDF file, create the indexes, and save the memory image file containing the internal data structures to disk.
- “BRAHMS load image” or “BRAHMS” – warm-start of BRAHMS: working with previously created memory image file.

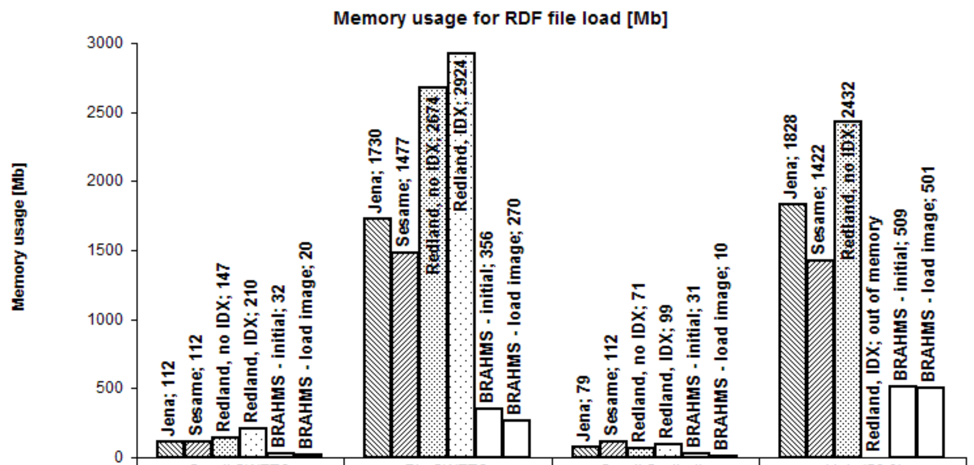


	Small SWETO	Big SWETO	Small Synthetic	Univ (50,0)
▧ Jena	19	353	12	1571
▨ Sesame	8	158	7	2820
▩ Redland, no IDX	7	179	4	258
▦ Redland, IDX	7	524	5	0
□ BRAHMS - initial	5	363	3	1256
□ BRAHMS - load image	0.1	2	0.1	3

Figure 3.14 Ontology load times for testes storages

Loading times for smaller datasets did not reveal significant differences, as main-memory implementation of all storages are specifically designed for handling small ontologies. For many prototype testing purposes, using an ontology with the size of a few Mbytes may be sufficient. However, bigger datasets revealed significant differences in loading times. The differences are in orders of magnitude. The parsing of the RDF file and creating the necessary indexes is computationally expensive. Utilizing the approach with a memory snapshot in BRAHMS allows us to compute them only once during the cold-start of the system, and later reuse the pre-calculated ontology image in experiments. The created ontology image consists of a contiguous memory array, divided logically into appropriate tables and indexes. Such organization of a snapshot as a single memory array allows us either to load the whole snapshot into main memory or to use it as a memory mapped file. In case of loading a snapshot to exclusive memory, the warm-start requires the time needed by the system to read the whole file from disk. When a memory mapped file option is used, the warm-start of the system is almost instantaneous, as all headers defining logical structure of the storage are located at the beginning of the file. Indexes, values and statements are loaded on demand and access is managed by operating system.

Along with measuring the parse and load time for the used ontologies, we measured the memory usage for each of the tested systems. The results are presented in Figure 3.15.



	Small SWETO	Big SWETO	Small Synthetic	Univ (50,0)
▧ Jena	112	1730	79	1828
▨ Sesame	112	1477	112	1422
▦ Redland, no IDX	147	2674	71	2432
□ Redland, IDX	210	2924	99	0
□ BRAHMS - initial	32	356	31	509
□ BRAHMS - load image	20	270	10	501

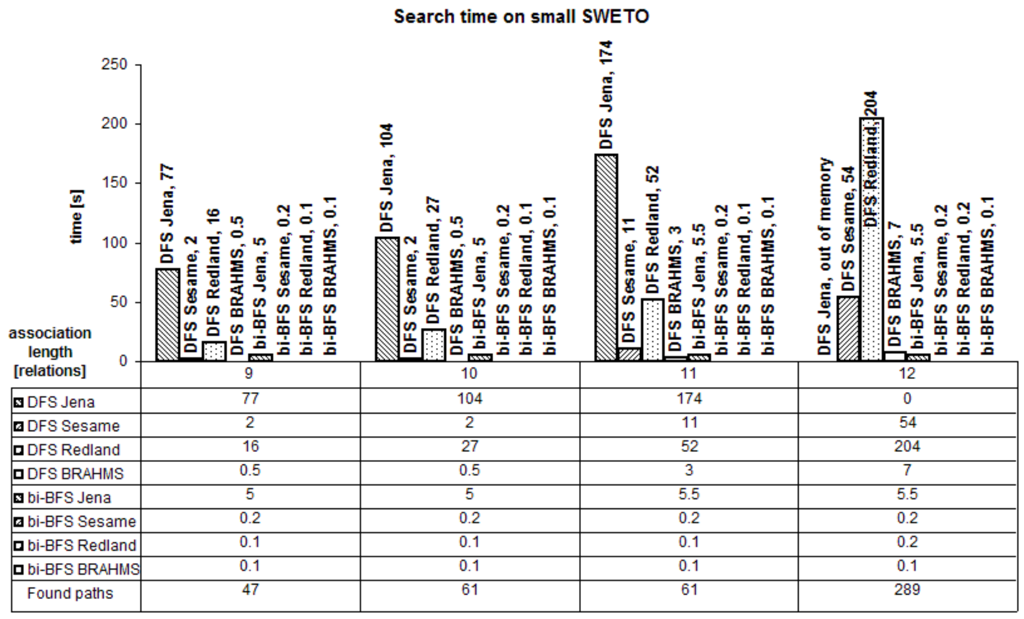
Figure 3.15 Ontology memory footprint in testes storages

The memory footprints for small ontologies show differences across the tested systems, but the memory requirements in range up to 150Mb may be insignificant for the majority of hardware configurations available today. For large datasets, the memory requirements become an important issue that can significantly limit the maximum size of the used ontology. In the performed ontology load tests, Jena, Sesame, and Redland used significantly more memory than BRAHMS for storing the same datasets. Both Redland implementations almost reached the hard limit of the available main-memory, and in one case, Redland was unable to load the dataset due to the lack of the available memory. Optimization of memory structures for BRAHMS resulted in a substantial reduction of memory requirements, which was especially noticeable for large ontologies. The compact, in-memory representation allows BRAHMS to work with larger datasets and leave more free memory for running the necessary graph algorithms.

3.3.4. Semantic association discovery experiments

In this section, we present the results from running semantic association algorithms based on DFS and bi-BFS on different datasets using Jena, Sesame, Redland, and BRAHMS. In the performed tests, we have used Redland patched with two additional indexes to achieve its highest performance. BRAHMS was using the previously prepared memory snapshot file. The timing results included only the algorithm execution time, while loading and parsing time were excluded.

A reasonable size of the small SWETO ontology enabled us to run performance tests using both algorithms on all storages. We present the timing results of discovery of semantic association up to length 9, 10, 11, and 12, in Figure 3.16.

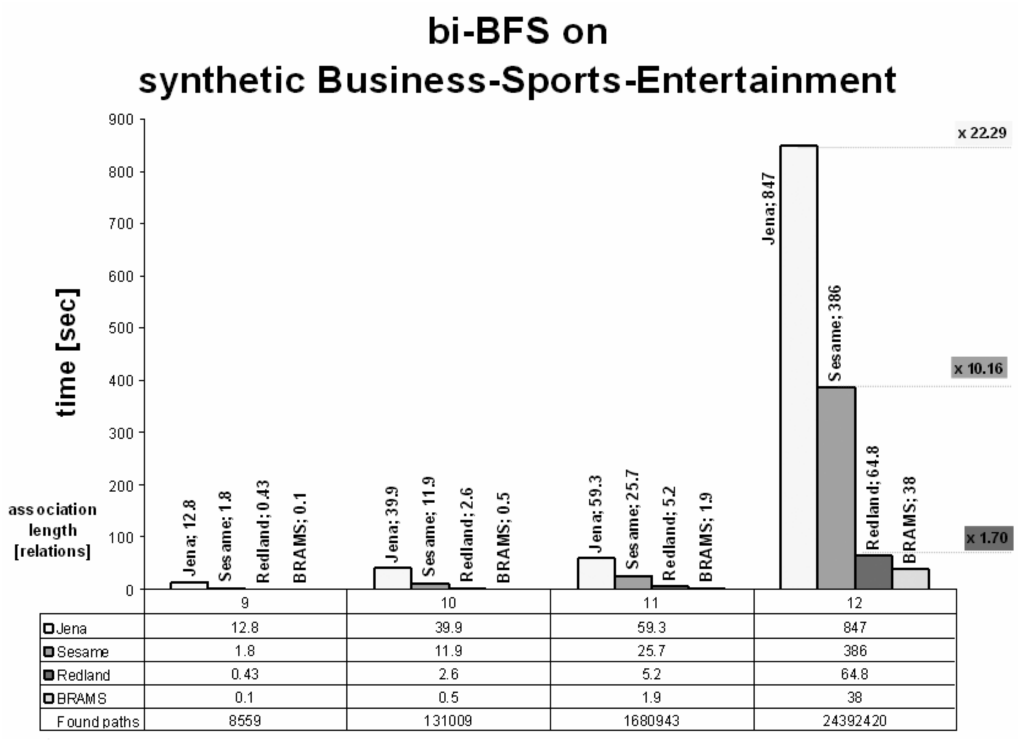


association length [relations]	9	10	11	12
▨ DFS Jena	77	104	174	0
▨ DFS Sesame	2	2	11	54
◻ DFS Redland	16	27	52	204
◻ DFS BRAHMS	0.5	0.5	3	7
▨ bi-BFS Jena	5	5	5.5	5.5
▨ bi-BFS Sesame	0.2	0.2	0.2	0.2
◻ bi-BFS Redland	0.1	0.1	0.1	0.2
◻ bi-BFS BRAHMS	0.1	0.1	0.1	0.1
Found paths	47	61	61	289

Figure 3.16 Timing results of DFS and bi-BFS on small SWETO

The DFS-based association discovery algorithm is significantly slower than the bi-BFS version. The DFS-based association discovery algorithm can return results in reasonable time for discovering shorter associations. For longer semantic associations or a highly connected graph, the time required by the DFS algorithm to complete may be unacceptable. Bi-BFS is an order of magnitude faster, but the speed-up comes at the price of high memory requirement.

The synthetic graph generated using TOntoGen has different connectivity characteristics than the real-life SWETO. The number of neighbors for each entity is distributed more evenly and hotspot nodes are relatively rare. A hotspot is a node with a large number of neighbors, which is significantly higher than the average number of neighbors in the graph. These features imply that the number of located simple paths between entities is significantly larger than in real-life datasets, and the algorithm performs relatively slowly.

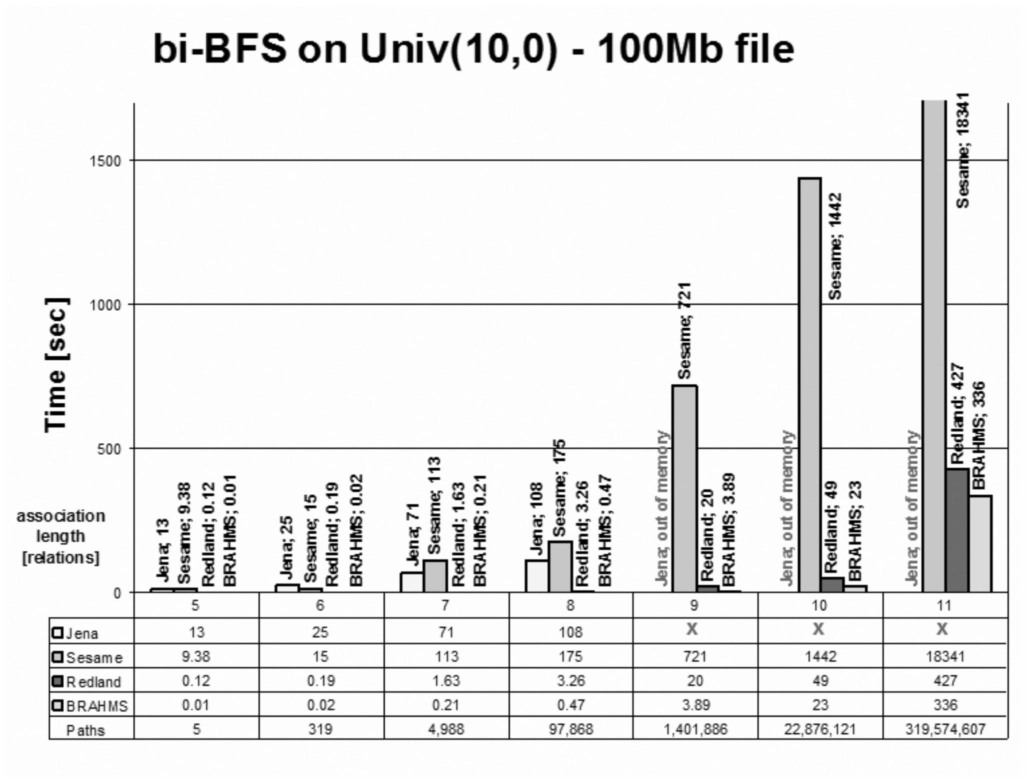


	9	10	11	12
□ Jena	12.8	39.9	59.3	847
▨ Sesame	1.8	11.9	25.7	386
■ Redland	0.43	2.6	5.2	64.8
□ BRAMS	0.1	0.5	1.9	38
Found paths	8559	131009	1680943	24392420

Figure 3.17 Timing results of bi-BFS algorithm on TOntoGen synthetic graph

The search for semantic associations of length up to 12 in TOntoGen resulted in over 24 million simple paths, while in the small SWETO, the dataset of similar size, there were only 289. All of the tested storages were able to perform the search, but the time required for calculating the results for longer paths differed significantly. BRAHMS proved to be the fastest.

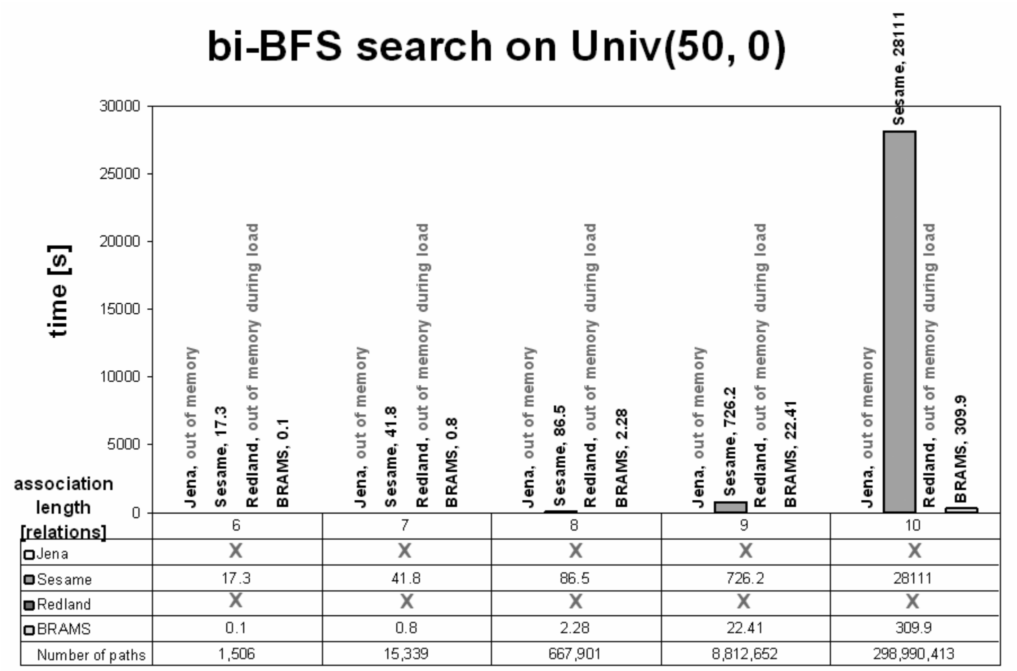
Finally, we performed scalability tests using Lehigh University Benchmark datasets of different sizes. The goal of this experiment was to test what sizes of ontology are feasible for each of the tested storages, and how efficiently semantic associations can be calculated, as the size of a dataset increases. We present the results for Univ(10,0), Univ(50,0) and Univ(700,0).



	5	6	7	8	9	10	11
□ Jena	13	25	71	108	X	X	X
▨ Sesame	9.38	15	113	175	721	1442	18341
■ Redland	0.12	0.19	1.63	3.26	20	49	427
□ BRAHMS	0.01	0.02	0.21	0.47	3.89	23	336
Paths	5	319	4,988	97,868	1,401,886	22,876,121	319,574,607

Figure 3.18 Bi-BFS timing results on Univ(10,0) dataset

All tested ontology storages were able to perform semantic association discovery up to length 8 on Univ(10,0) dataset in reasonable time. For longer association paths, Jena ran out of memory and failed the search, while the execution time on Sesame became almost unacceptable. Both Redland and BRAHMS were able to provide results in a comparable time. The search for longest associations (up to length 12) returned almost 320 million simple paths connecting the two selected entities.

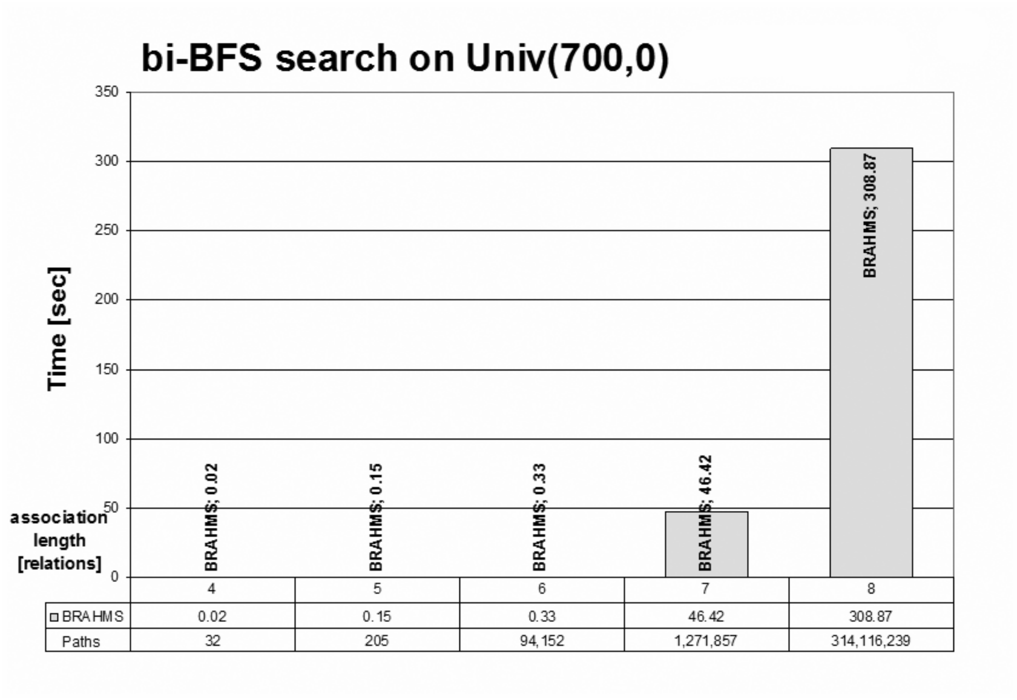


association length [relations]	6	7	8	9	10
□ Jena	X	X	X	X	X
▨ Sesame	17.3	41.8	86.5	726.2	28111
■ Redland	X	X	X	X	X
□ BRAMS	0.1	0.8	2.28	22.41	309.9
Number of paths	1,506	15,339	667,901	8,812,652	298,990,413

Figure 3.19 Bi-BFS timing results on Univ(50,0) dataset

The number of semantic associations increased dramatically with the increase of the dataset size to Univ(50,0). In this experiment, there were almost 300 million associations found for the length of only 10, while in the experiments with smaller dataset, Univ(10,0), we reached similar number of discovered paths for associations of length 12. Jena and Redland could not perform the search for semantic associations even up to length 6, due to either a lack of available memory, or inability to load the dataset itself. While the tests of Sesame storage completed, the performance results were far behind those achieved by BRAHMS.

The final experiment including Univ(700,0) was performed on a 64bit machine with 8Gb of RAM and included only the BRAHMS storage. We did not include results from other storages in this test, as they did not manage to load such a large ontology using their main-memory implementation. We attempted to test the limits of the ontology size that could be handled by BRAHMS for associations discovery.



	4	5	6	7	8
□ BRAHMS	0.02	0.15	0.33	46.42	308.87
Paths	32	205	94,152	1,271,857	314,116,239

Figure 3.20 Bi-BFS timing results on Univ(700,0) dataset

BRAHMS successfully created the memory snapshot of such a large ontology and was able to perform searches for semantic associations. Within the ontology of this size, we were only able to reach associations of length up to 8 relationships. For longer paths, there was not enough memory to allocate for the necessary structures to perform the search.

In all of the performed tests, we have concentrated both on the speed and on the memory requirements. BRAHMS proved to have a very compact and efficient main-memory representation of the ontology. These features allowed us to effectively search for long semantic associations even in very large graphs, despite the fact that search for simple paths in graph is an NP-Complete problem. Using BRAHMS, we are able to effectively find hundreds of millions of associations in a reasonable time. The number of all possible associations can be overwhelming, as we increase the search distance. The next and very important step to finding interesting semantic associations is to include a selectivity of the associations based on their semantics, specific patterns, or desired importance. We describe SPARQLeR, our solution and a prototype implementation of specifying interesting semantic associations using regular expressions, in the next chapter.

CHAPTER 4 ADVANCED ONTOLOGY QUERYING: SPARQLER

In this chapter, we present SPARQLeR, a novel extension of the SPARQL query language with path queries of flexible length. It allows performing the semantic association discovery using a high-level query language over the BRAHMS ontology storage. We propose the use of regular expressions for defining the semantically relevant path patterns that have to be included in searched association. The fine-granularity of path properties is achieved by restricting types, individual entities, and relationships included in the path. The proposed extension fits seamlessly into the current syntax and philosophy of SPARQL. We provide a detail description of the proposed extension together with many query examples. Finally, we present the test results of the working prototype of SPARQLeR implemented using the BRAHMS ontology storage.

4.1. Desired features for discovery of interesting semantic associations

Semantic association, as defined in Section 3.1 is an undirected simple path that connects two entities in the knowledge base. We demonstrated in the previous chapter that location of all the semantic associations up to a specific length between a pair of selected entities is possible in an on-line time (in most cases), even in very large ontologies. The presented algorithms are able to efficiently locate hundreds of millions of simple paths between entities, where each of them is a meaningful semantic association. Consequently, with the use of the previously proposed algorithms, a user can easily be flooded with enormous number of correct results, while the really interesting ones from the user's perspective contribute only a small fraction of them.

A selective approach for locating semantic associations interesting to the user requires a level of language support that allows a definition of certain restrictions, which can be imposed on the located paths. The proposed algorithms do not facilitate searching for the restricted types of associations, which have certain properties, entities, features, and may include specific patterns. The current ontology query languages, as presented in Section 3.1.1, also do not have such support, as they are limited to fixed graph patterns with allowed optional clauses. We believe that querying for semantic associations is an important feature missing in the current RDF query languages. In SPARQLeR, we propose an extension of the SPARQL query language that allows semantic association discovery. In the following sections, we present the essential requirements we think should be included in the definition of path restrictions.

4.1.1. Repeating patterns of properties

Repeating patterns of properties in association paths are common in biological domains. For example, KEGG [59] contains information on hundreds of known metabolic pathways. They describe metabolic processes within organisms and consist of chains of chemical reactions. A lot of them intersect, as they share substrates, catalysts, and products. Biological compounds have also additional associations. They may include their location in the cell, specific organism, defined functionality, and other. All these features create a complex network of relationships where locating and retrieving of metabolic pathways becomes a difficult problem.

A metabolic pathway consists of a sequence of chemical reactions. It can be interpreted as a repeating pattern of incoming substrates and outgoing products. Concentrating the search only on such regular patterns can be used to locating semantic associations that describe only metabolic pathways. Using specific queries with these restrictions, scientists should be able to query for and retrieve ordered sequences of specific reactions that lead from a given substance to a desired final product.

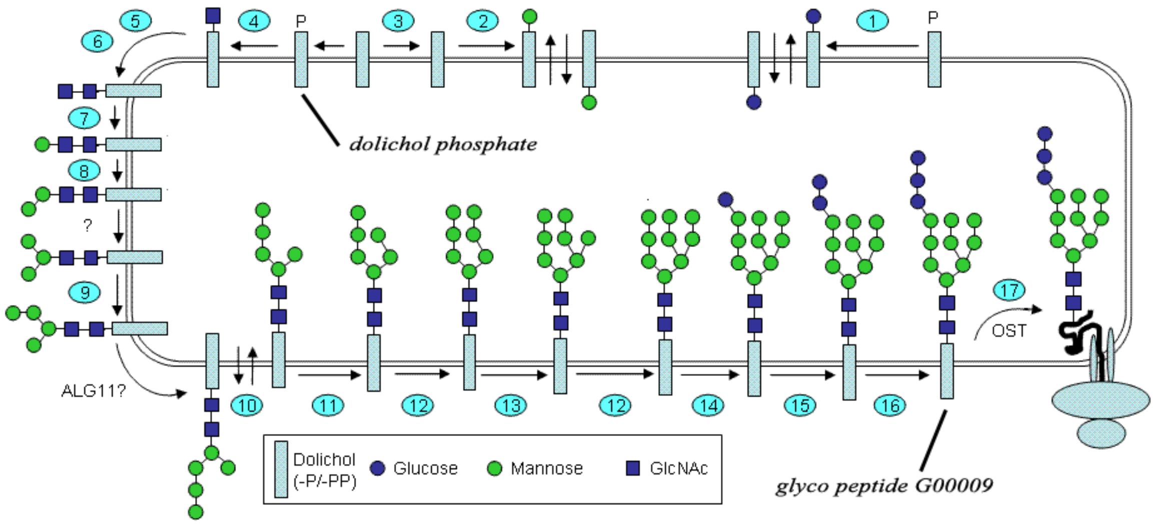


Figure 4.1 N-Glycan biosynthesis pathway
(courtesy of Dr. Alison Vandersall-Nairn, University of Georgia)

A well-known metabolic pathway in glycobiology, an N-Glycan Biosynthesis pathway [49], is presented in Figure 4.1. It starts from *dolichol phosphate* and ends with the production of *glyco peptide G00009*. It includes only 15 chemical reactions represented by arrows, and is considered a relatively a short pathway in biology. However, in the area of Semantic Web, a path of a length 15 relationships is considered very long. As demonstrated in the previous chapter, locating simple paths up to a certain length is feasible even in very large ontologies, although the paths of such length were already too long to locate. Therefore, the proposed solution for searching of semantic associations with specific patterns cannot be simply implemented as a selection within results produced by the previous algorithms, but would require selecting only interesting patterns while performing the search.

4.1.2. Semantic association as an undirected path

A directed path representing a sequence of chemical reactions, as shown in the previous example, naturally captures the semantic association between *dolichol phosphate* and *glyco peptide G00009*. Let us use a slightly different representation of a chemical reaction in the ontology. In this model, each reaction becomes a first class object and it has associated substrates and products. For illustration, let us present in Figure 4.2 a short fragment of the N-Glycan biosynthesis pathway from the GlycO and EnzyO ontologies [91]. The shown fragment includes three reactions without any additional properties present in the ontology, which can be mapped to reactions No 6, 7 and 8 in Figure 4.1. The reactions are represented by the entities R05972, R05973, and R06238 and glycans G00002, G00003, G00004, and G00005 are entities associated with reactions as their reactants and products.

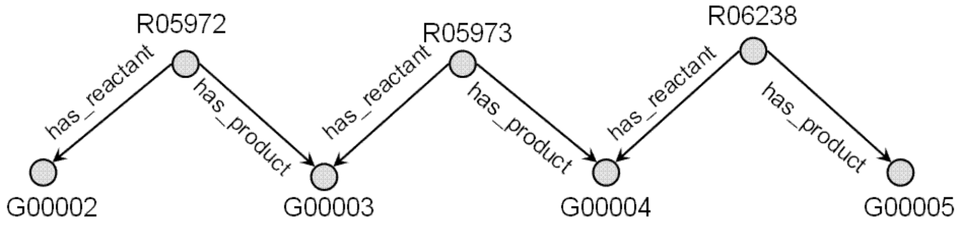


Figure 4.2 Fragment of N-Glycan biosynthesis pathway from GlycO ontology

Glycan G00002 is a predecessor of G00005 and although there is no directed path connecting them in the ontology, they are semantically related in a meaningful way. In this ontology, the sequence of reactions between *dolichol phosphate* and *glyco peptide G00009* is modeled as an undirected path. Using the representation of reactions as first class objects, this semantic association becomes 30 relationship long.

4.1.3. Restrictions on resources in the path

Many metabolic pathways are not just a simple sequence of reactions. In several cases, there exist alternative reactions leading to the same product. As the pathway branches out and later joins, scientists may be interested in investigating only a specific branch of a pathway, the one that includes a given reaction or an intermediate product.

In the example of N-Glycan biosynthesis pathway in Figure 4.1, there are two alternative reactions: No 5 and No 6. In the search for specific semantic association we should be able to express, that we are interested only in metabolic pathways between *dolichol phosphate* and *glyco peptide G00009* that include reaction No. 6, represented by the entity R05969 in the ontology. Similar restrictions should be allowed to be imposed on entities included in the path.

4.1.4. Relationship directionality on the path

In graph theory, if there exists a path connecting two vertices in a labeled directed graph it can be either a directed or an undirected path. For capturing a specific semantics of an association, the use of a directed path may be too restrictive. On the other hand, allowing undirected paths can be too undefined, and lose the desired semantics. The directionality of each relationship in the path defines a specific meaning. Following the direction of a named relationship connecting resources A and B should be interpreted accordingly to its label, while the reverse directionality should be treated as its inverse. In theory, each named relationship in an ontology has its inverse and if there is a path connecting two resources, a directed path can always be composed of the defined relationships or their inverses. In practice, although the definition of an inverse relationship is allowed, ontologies rarely use this feature and thus the proposed extension should not rely on it. Query languages, which allow defining a graph pattern by a conjunction of individual statements do not require the use of inverse relationships. In contrast, a definition of a path consists of a sequence of statements. For this reason, both relationships and their inverses should be allowed in the definition of a path pattern.

Spatial relationships, such as *A is inside B*, offer illustrative examples for understanding the meaning of directionality of relationships included in the path. Let us consider the following three path queries, expressed as regular patterns:

1. *spatial:inside** - when used in a search for directed paths, it locates semantic associations illustrated by a diagram shown in Figure 4.3a.
2. *spatial:inside** - when used in a search for undirected paths, it locates semantic associations illustrated by diagrams shown in Figure 4.3a, b and c.
3. (*spatial:inside spatial:inside*$^{-1}$)* [read as: concatenation of inside with inverse of inside] - when used in a search for directed paths, it locates semantic associations illustrated by a diagram shown in Figure 4.3c, showing very specific, a chain-like inclusion structure.

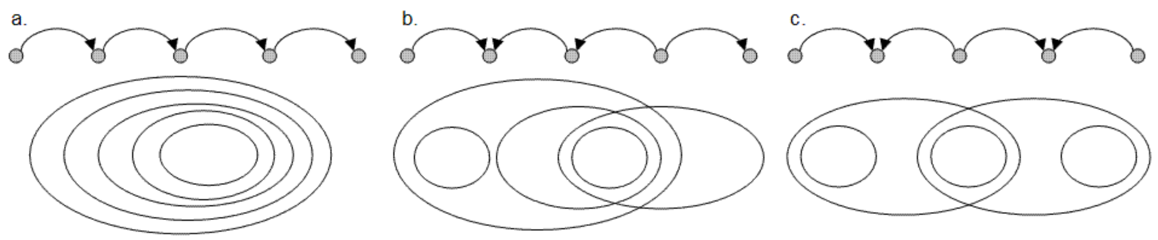


Figure 4.3 Examples and set interpretation of spatial queries

Without the use of the inverse property in the path pattern definition, obtaining a specific result such as the one presented in Figure 4.3c would not be possible. Defined directionality of the relationships included in the path provides a unique semantic interpretation of an association. Following the above observation, we believe that the proposed extension for the discovery of semantic associations should allow a more precise definition of a path than only directed or undirected ones.

4.2. SPARQLeR definitions and syntax

In this section, we present the details of SPARQLeR, an extension of the SPARQL query language for semantic association discovery using regular path patterns. Our intension has been to introduce only minimal changes to the SPARQL's syntax and semantics, while providing maximum flexibility in defining path patterns and the restrictions. The new constructs in SPARQLeR are designed for the discovery of semantic associations and, in particular, allow the user to:

- easily combine path searches with statement-based graph patterns,
- create path patterns between two resources, as well as patterns for exploratory single-source paths,
- restrict located paths by requiring the presence of specific resources on the path, possibly even at a specific position,
- search for paths with a specific directionality of properties, where the use of the inverse of property is allowed,
- precisely describe the path semantics with the use of regular expressions formed over properties included in the path,
- specify that a path be directed or undirected,
- specify types of elements (instances, schema classes, literals) that are allowed to be included in the path,
- indicate if the hierarchy of sub-properties should be used in property matching, and

- impose a lower and upper limit on the length of located paths.

4.2.1. Path – a meta-property

SPARQL graph patterns are built from conjunction of triples. In order to fit seamlessly into the current SPARQL grammar, we define a path as a meta-property. This allows a natural use of triples, where a path is treated as a normal property connecting two resources in constructing more complex graph patterns.

To formally define a path as a *meta-property* in the RDF description base, we have created a new class called *Path* in the new vocabulary called rdf-meta-schema. The class *Path* has been defined as the sub-class of both *rdf:Property* and *rdf:Seq* as follows:

```
<rdf:Class rdf:about="http://meta.org/rdf-meta-schema#Path">
  <rdfs:isDefinedBy
    rdf:resource="http://meta.org/rdf-meta-schema#"/>
  <rdfs:subClassOf
    rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:subClassOf
    rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"/>
  <rdfs:label>Path</rdfs:label>
  <rdfs:comment>The class of RDFMS paths.</rdfs:comment>
</rdf:Class>
```

Figure 4.4 Definition of *Path* class as a meta-property in the RDF meta schema

A resource of type *Path* is both a property and a sequence of other RDF resources. We further restrict the above definition to fit the requirements imposed by the definition of semantic association. As defined in Section 3.1, a semantic association is a simple undirected path connecting two resources in an RDF description base. Similarly, a *Path* is composed of an ordered list of entities and connecting them properties of set directionality. This is achieved by defining *Path* as a subclass of *rdf:Seq*. The *Path* class is also defined as a subclass of *rdf:Property*, and can be legally used to connect two resources. For this reason, only a property type resource is allowed both on the first and the last positions in the sequence of resources.

The provided definition of the *Path* can serve both as a semantic association, and as a normal property in an RDF description base. Two resources A and B in RDF connected with the defined meta-property, can be treated as a *meta-triple*. This extends the applicability of semantic associations to the RDF description base, while keeping the grammar of constructing graph patterns with triples.

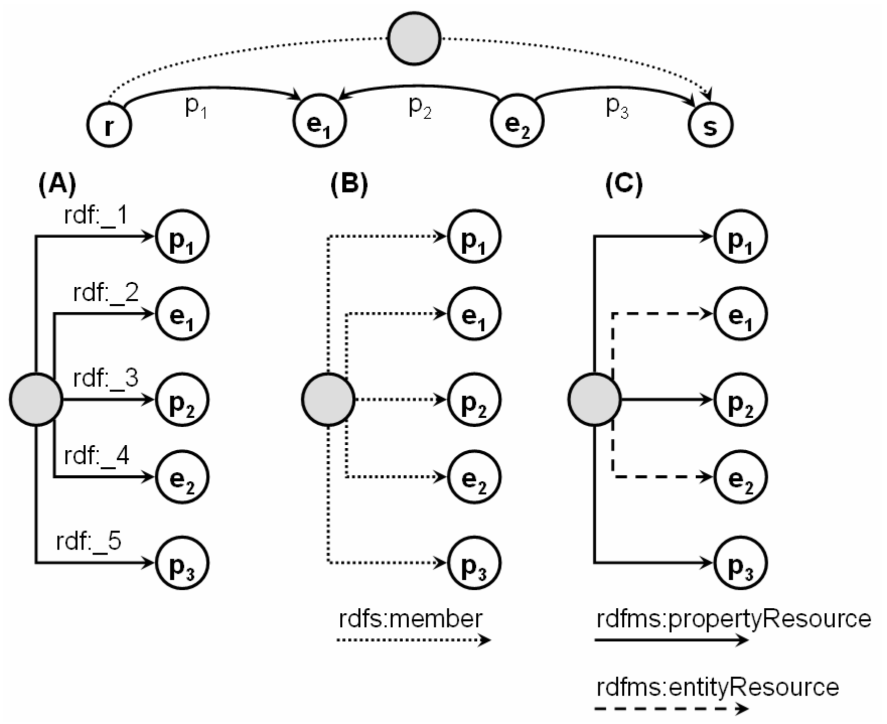


Figure 4.5 Instance of a *Path* class – a meta-property and a sequence

An instance of *Path* type, indicated by a shaded circle, is presented in Figure 4.5. It represents a semantic association between resources *r* and *s*. By the *Path* class duality incorporated in its definition, it can be seen as a meta-property (upper part) or a sequence (lower part). If it is interpreted as a sequence, members of the path can be accessed using variety of relationships. According to *rdf:Seq* definition, all elements can be reached using the *rdfs:member* property (B), and additionally each of the elements can be accessed individually by specifying its position in the sequence (A). Properties and entities, which appear interchangeably in the path, can be accessed using the two newly defined properties in the meta-schema (C). The properties and examples of their use are described in the forthcoming sections of this chapter.

4.2.2. Basic path patterns

Semantic association discovery in SPARQL is supported by path queries. A *path query*, is a query in SPARQLeR that contains at least one path pattern. A *path pattern* is a triple pattern in SPARQLeR that matches a *Path* meta-property between two resources. Instead of a normal variable in place of a property, it has a newly defined *path variable*, so that the new constructs fit seamlessly into the current SPARQL syntax. To distinguish between normal variables in SPARQL and path variables, path variables are represented by names starting with the percent character (**%**). For example, *?x* is a normal variable, and *%assoc* is a path variable.

One of the basic path patterns involves searching for all directed paths between two selected resources:

```
(1)  SELECT %path WHERE {<r> %path <s>}
```

The presented SELECT path query (1) matches all directed paths between the resources *r* and *s*. Directed path is set as a default matching for a path query. Every located path is bound to the *%path* variable as expected from the SPARQL semantics. A located path is a sequence of properties and entities, and since it is an instance of a container class (*rdf:Seq*) it is represented by an RDF blank node. Therefore, opposite to intuitive expectation, the result of this query is a list of blank nodes representing the sequences of located paths, and not a list of their individual elements.

The new *list* operator has been introduced for providing all properties and entities contained on the path. It is applicable only to path variables and lists the elements of each located path.

```
(2)  SELECT list(%path) WHERE {<r> %path <s>}
```

This SELECT path query (2) locates all directed paths between resources *r* and *s*, and for each located path it lists all matched properties and entities.

Another extension of SPARQLeR is a more exploratory query used for semantic association represented by a single source path.

```
(3)  SELECT %path, ?res WHERE {<r> %path ?res}
```

The following single source SELECT path query (3) finds all the resources reachable by a directed path from a given resource *r* together with the paths themselves. Each result is a pair formed by the reached resource, bounded to *?res* and a path *%path* leading to it. The analogous form of the above query relies on the inverse path pattern of the form `{?res %path <r>}`. This pattern matches all resources (and paths) from which the resource *r* is reachable by a simple directed path.

We strictly limit path queries to double source and single source path patterns. We require that at least a subject or an object in the path triple pattern is either a specific resource instance, or a variable used as a subject or object in another regular (non-path) triple. This limitation is imposed due to fact, that searching for unbounded regular paths in a graph presents a different class of a problem, which has a much higher complexity and is not handled by our algorithms.

4.2.3. Restrictions on elements in the path

In the path queries, we treat a path variable as a meta-property that connects two resources in the graph. For defining restrictions on the included resources in the located path we can utilize the fact that a path pattern is also defined as a sequence of resources. The RDF/S vocabulary provides two properties we can directly apply here: *rdfs:member* and *rdf:_nnn*.

In the select path queries, we can require that the located path include a specific entity. It is achieved by interpreting a path as a sequence and testing the entity membership (as described in Section 4.1.3). An example of such query is presented below.

```
(4)  SELECT %path WHERE
     {<r> %path <s> .  %path rdfs:member <e>}
```

The query (4) matches any directed semantic path between the resources *r* and *s*, provided the path includes the resource *e*. The resource *e* can be at any position in the sequence, and as a path pattern includes entities and properties, it can be either of them. The query (4) combines the path pattern with a normal triple. Although *%path* variable is used in the second triple, it is interpreted as sequence (*rdf:Seq*), as it is not in the place of a property in the triple.

```
(5)  SELECT %path WHERE
     {<r> %path <s> .  %path rdf:_1 <p>}
```

The path query (5) matches all directed paths between the resources *r* and *s*, provided the path begins with the property *p*. The property *rdfs:_2* can be used to match the first entity on the path, assuming that it exists.

A path in SPARQLeR always begins and ends with a property, and the number of elements in the path sequence is odd. The resources at odd positions in the sequence are properties, and even positions are occupied by entities. Fully exploiting the potential of a specific definition of a path in SPARQLeR requires testing of only odd or only even positions in the path sequence. However, the standard RDF/S vocabulary does not include suitable relationships. To alleviate this problem, we have defined two additional properties in the meta-schema: *rdfms:entityResource* and *rdfms:propertyResource*. They allow testing if a given resource is at an entity position in the sequence (even) or if it occupies a property spot (odd position).

```
(6)  SELECT %path WHERE
     {<r> %path <s> .  %path rdfms:entityResource <e>}
```

The select path query (6) returns only these directed paths between resource *r* and *s* that go through entity *e*. In this query resource *e* has to be on the even place in the located path and therefore cannot be of an *rdf:Property* type.

```
(7)  SELECT %path WHERE
     {<r> %path <s> .  %path rdfms:propertyResource <p>}
```

The select path query (7) returns only these directed paths between resources *r* and *s* that include property *p*. In this query, the resource *p* has to be at the odd position in the located path and therefore has to be of the *rdf:Property* type.

4.2.4. Regular expressions, defined directionality and property inverse

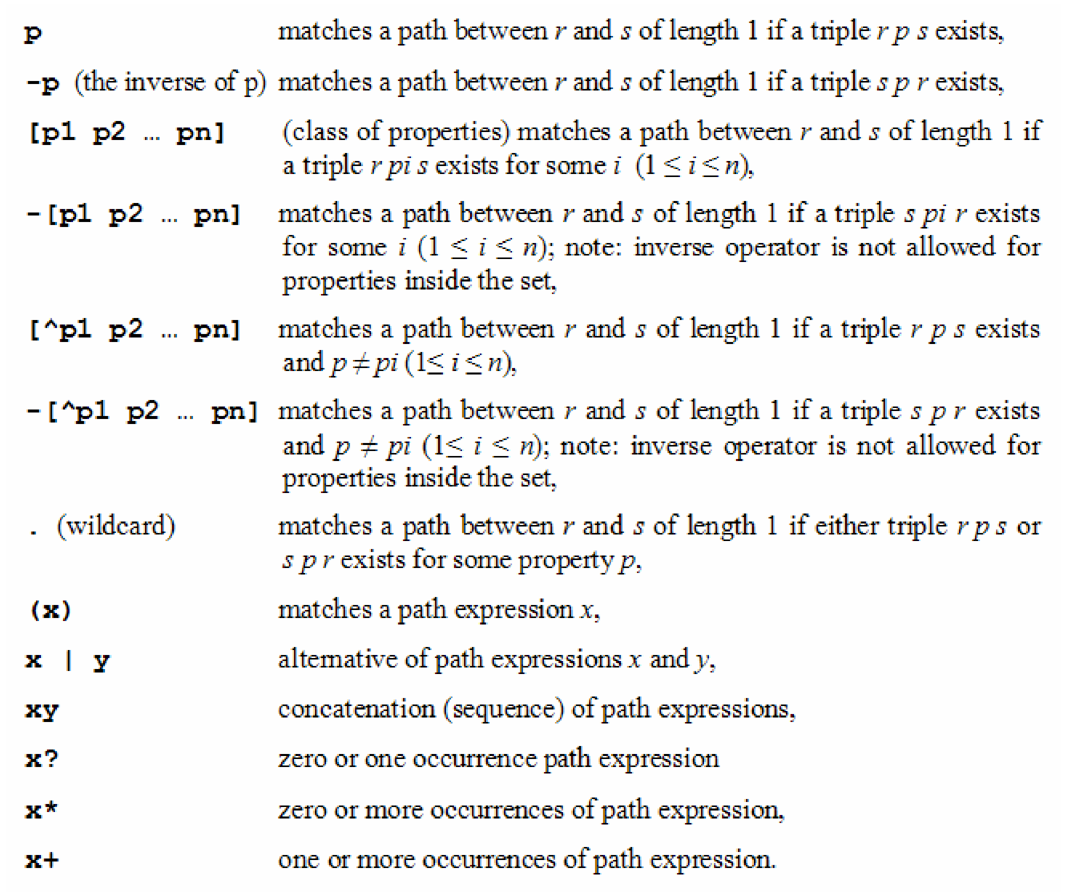
SPARQLeR allows building path patterns using regular expressions over properties included in the path. Path patterns are not limited only to directed and undirected paths. As presented in Section 4.1.4, there is a need for expressing the semantics of a *defined directionality path*. Graph patterns in SPARQL include only simple triples and therefore there is no need for a property inverse. The definition of a path requires stating explicitly the directionality of included properties. In SPARQLeR, we use the '−' (minus) character to denote the inverse of a property.

Regular expressions are supported in SPARQL within the FILTER clause by the *regex* operator. We decided to utilize the syntax of the existing operator and extend it with the support for regular path expressions. It is used for testing regular path expressions on path variables in a path query.

In the extended *regex* operator, the first argument is a path variable, the second argument describes a regular path expression over the properties included in the path, and the third argument includes optional flags for path matching. The full invocation syntax is as follows:

```
regex( pathvar, pathexpr, pathflags )
```

The path expressions allow specifying an explicit semantics of a searched path. They can be formed with the use of property names, their inverses, classes of properties, and the usual collection of regular expression operators. Assuming a path pattern *<r> %path <s>* and a filter expression *regex(%path, pathexpr)*, we recursively define paths between the resources *r* and *s* which are matched by the defined path expression *pathexpr*, according to the composition on the path expression. p, p_1, p_2,…, and p_n denote properties in the ontology, and x and y represent path expressions. SPARQLeR's path expressions and their interpretation is presented in Figure 4.6.



`p`	matches a path between *r* and *s* of length 1 if a triple *r p s* exists,
`-p` (the inverse of p)	matches a path between *r* and *s* of length 1 if a triple *s p r* exists,
`[p1 p2 ... pn]`	(class of properties) matches a path between *r* and *s* of length 1 if a triple *r pi s* exists for some *i* $(1 \le i \le n)$,
`-[p1 p2 ... pn]`	matches a path between *r* and *s* of length 1 if a triple *s pi r* exists for some *i* $(1 \le i \le n)$; note: inverse operator is not allowed for properties inside the set,
`[^p1 p2 ... pn]`	matches a path between *r* and *s* of length 1 if a triple *r p s* exists and $p \ne pi$ $(1 \le i \le n)$,
`-[^p1 p2 ... pn]`	matches a path between *r* and *s* of length 1 if a triple *s p r* exists and $p \ne pi$ $(1 \le i \le n)$; note: inverse operator is not allowed for properties inside the set,
`.` (wildcard)	matches a path between *r* and *s* of length 1 if either triple *r p s* or *s p r* exists for some property *p*,
`(x)`	matches a path expression *x*,
`x | y`	alternative of path expressions *x* and *y*,
`xy`	concatenation (sequence) of path expressions,
`x?`	zero or one occurrence path expression
`x*`	zero or more occurrences of path expression,
`x+`	one or more occurrences of path expression.

Figure 4.6 Definition of allowed regular expressions in path definition

Regular path expressions in SPARQLeR are defined on properties. They are used to form a specific path pattern with explicitly defined semantics. Examples of regular path expressions and their graphical interpretation as directional paths are presented in Figure 4.7.

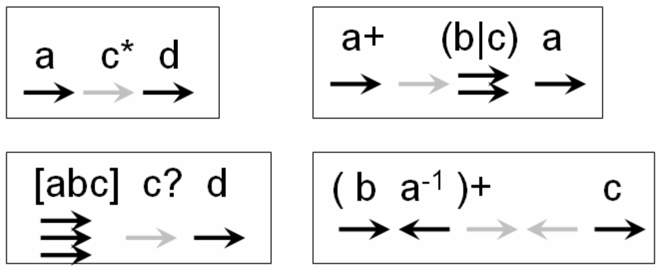


Figure 4.7 Regular path expressions examples and matching paths

Regular path expressions can be defined in the FILTER clause in the query. For example, the following SPARQLeR query (8) matches all directed paths between resources *r* and *s* that use only property *foo:prop* of length one or more:

```
(8)   SELECT list(%path) WHERE
      { <r> %path <s>
      FILTER( regex(%path,"foo:prop+") }
```

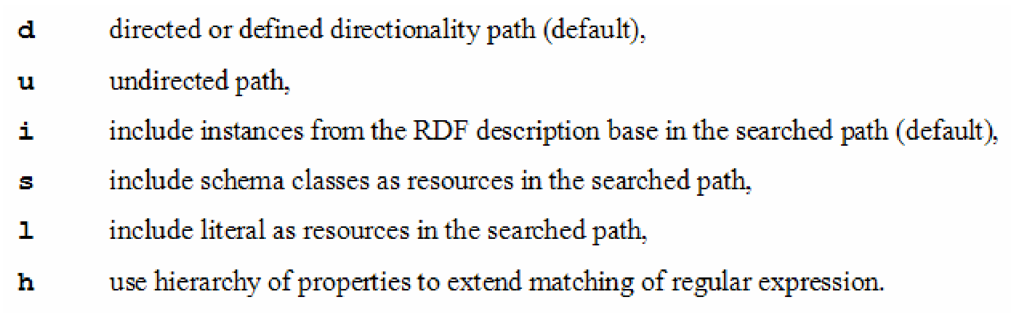
A more complex example of a regular path expression is presented below:

```
(9)   SELECT list(%path) WHERE
      {<r> %path <s>
      FILTER(regex(%path,"foo:p1 ((foo:p2|foo:p3) -foo:p4 )+") }
```

In the select query (9), a located path must start with a property *foo:p1*, and then a pattern of *foo:p2* or *foo:p3,* followed by the inverse of *foo:p4* must be repeated at least once. This expression matches associations with the defined meaning (properties), that are of an odd length of at least 3 relationships.

4.2.5. Path pattern flags

In the third argument, the *regex* expression syntax allows as specifying optional flags used for matching. We use it for specifying the directionality of the path (directed or undirected), listing included types of resources in the path, and specifying if a property hierarchy should be used in evaluation of a regular expression. The flags are specified by using designated letters in the flags string. The following flags are supported in the regular path expressions:



d	directed or defined directionality path (default),
u	undirected path,
i	include instances from the RDF description base in the searched path (default),
s	include schema classes as resources in the searched path,
l	include literal as resources in the searched path,
h	use hierarchy of properties to extend matching of regular expression.

Figure 4.8 List of supported flags for regular paths expressions

The default setting is "**di**" which specifies searching for defined directionality paths that include only instances from the RDF description base, and do not use the property hierarchy in matching. Flags "**d**" and "**u**" are mutually exclusive, but can be combined with any of the remaining flags.

For example, the following select query (10):

```
(10) SELECT list(%path) WHERE
     {<r> %path <s>
     FILTER(regex(%path,"[foo:p1 foo:p2 rdf:type]+","uis") }
```

searches for undirected paths that include both instance entities and schema classes in the path. Each located path must include only relationships *foo:p1*, *foo:p2*, and *rdf:type*, while the *rdf:type* relationship is used to allow the path to cross the boundary between the schema and the instance base.

The "**h**" flag allows us to exploit the hierarchy of properties in the ontology to expand the matching of path expressions. Let us assume that we have the following hierarchy of properties in the ontology: *foo:p11* and *foo:p12* are sub-properties of *foo:p1*, and *foo:p121* is a sub-property of *foo:p12*.

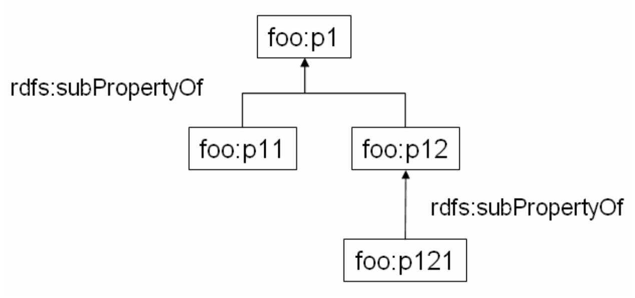


Figure 4.9 A hierarchy of properties

If we need to match a semantic association that includes the property *foo:p1* and all of its sub-properties in the hierarchy presented in Figure 4.9, we have an option of specifying all of them as a class of properties, or allow the property subsumption in the matching of path expressions. With the presented hierarchy of properties, the following two select queries (11) and (12) return the same results:

```
(11) SELECT list(%path) WHERE
     {<r> %path <s>
     FILTER(regex(%path,"foo:p1+","h") }

(12) SELECT list(%path) WHERE
     {<r> %path <s>
     FILTER(regex(%path,"[foo:p1 foo:p11 foo:p12 foo:p121]+")}
```

Wildcards that match any known property defined in the ontology are allowed in matching both directed and undirected paths. For example, the expression:

```
     regex(%path,".*foo:prop.*","ui")
```

specifies that the undirected path may include any known properties, but must involve the property *foo:prop* of any directionality (forward or inverse). Note, that in case of a directed path, *regex(%path,".*")* matches only known properties, even though the wildcard expression (.) matches both a property and its inverse.

4.2.6. Path length filter

The unlimited length of semantic association discovery is often a computational problem for the semantic association discovery algorithm. The fast algorithm based on bi-BFS is limited by the available memory for the expansion of search structures. On the other hand, the time complexity of the DFS algorithm, although in most cases is guaranteed to return results, may be too high.

The new *length* operator can be used to limit the length of semantic associations. The number of included properties in the path pattern is returned as the path length. The *length* operator can be used as part of a FILTER expression, where the path length can be compared to a constant value. For example, the select query (13),

```
(13) SELECT list(%path) WHERE
     {<r> %path <s>
     FILTER(length(%path)<5)}
```

restricts the matched directed paths to be of length less than 5.

Path filtering expressions may be combined with any other filter tests. In general, a select query may combine matching of a structural part of a query, specified with triple patterns, with filter expressions. An example of a more complex query is presented below:

```
(14) SELECT list(%path) WHERE
     {<r> %path <s> . %path rdfs:member ?x . ?x rdf:type <foo:A>
     FILTER(length(%path)<=6 && length(%path)>=4 &&
            regex(%path,"(foo:prop -foo:rel)+") }
```

The select query (14) matches a path (semantic association) between resources *r* and *s* that has the following properties:

- it is a directed path of length at least 4 and at most 6, where the relationships have a repeating pattern *foo:prop* and the inverse of *foo:rel*,
- the located path must contain an unknown entity (?x) that is of type *foo:A*.

4.2.7. Beyond the select queries

In addition to the SELECT queries, which are most commonly used, SPARQL includes the CONSTRUCT, DESCRIBE and ASK queries. Path patterns can be used in all of them, and we provide the specific semantics to interpret the results of such queries. CONSTRUCT, DESCRIBE and ASK queries have not been included in the prototype implementation, but for the completeness of the SPARQLeR extension, we provide a brief description of their semantics.

The CONSTRUCT queries are designed to form a new graph from the statements returned as the result set. Intuitively, using the located paths we should be able to build a valid graph. Let us define the result of using a path pattern in the construct query using following example:

```
(15) CONSTRUCT {<r> %path <s>}
     WHERE {<r> %path <s>}
```

The query (15) constructs a graph from statements included in all located directed paths that connect resources *r* and *s*. In construct queries, we do not allow the use of the *list* operator.

A combination of multiple path patterns in the CONSTRUCT query may lead to extracting semantically interesting subgraphs from the ontology. Let us take the following query as an example:

```
(16) CONSTRUCT {<r> %path1 <s> . <q> %path2 <t>}
     WHERE { <r> %path1 <s> . <q> %path2 <t> .
             %path1 rdfms:entityResource ?x .
             %path2 rdfms:entityResource ?x
     FILTER (length(%path1) <= 8 && length(%path2) <= 8) }
```

The result of the query (16) is a constructed graph that spans over four resources (*r*, *s*, *q*, and *t*) and the located semantic associations have at least one common resource denoted by *?x*. One of the applications of such query is as an alternative tool for the search of highly informative subgraphs, as described in [86].

The ASK query functionality for path patters is defined as testing for existence of at least one specified path. A sample query (17) for testing if there is a path between resource *r* and *s* up to length 5 is presented below.

```
(17) ASK %path
     WHERE { <r> %path <s>
     FILTER (length(%path) <= 5) }
```

The DESCRIBE query returns the description of all resources included in the located paths. A description of paths up to length 3, between resources *r* and *s* is the result of the following query (18):

```
(18) DESCRIBE %path
     WHERE { <r> %path <s>
     FILTER (length(%path) <= 3) }
```

4.3. Prototype implementation

Our prototype implementation of SPARQLeR uses BRAHMS, the high-performance main-memory RDF/S storage system presented in chapter 3. The path discovery algorithm is based on the previously tested bi-directional breadth first search (bi-BFS) algorithm, which in our previous tests proved to be the fastest practical solution for locating all simple paths between two resources in the ontology.

The implementation is built using specific *iterators* over statements from the lower-level BRAHMS API. A query pattern in SPARQL is transformed into a sequence of iterators to be executed during the search, based on the intersection points. A single path pattern is treated as a special type of a triple and requires creating an additional iterator.

The iterator for a path pattern combines the previously described discovery of simple paths with the restrictions imposed by the defined in Section 4.2.4 regular expression. It has been implemented as a hybrid of bi-BFS and a simulation of a Deterministic Finite Automaton (DFA) created for a given path expression. Naturally, Bi-BFS can include checking for path length limits imposed by including the *length* operator to the FILTER clause. The handling of regular expressions defined by the *regex* operator is done by the use of appropriate DFAs. Checking of any additional conditions, such as the entity or property membership in the path, is performed after the candidate path has been located.

Bi-BFS performs the search for semantic associations simultaneously in both directions. It searches for forward and reverse path fragments that can be later joined to create a single simple path. Creating a DFA that recognizes the regular language defined within the *regex* operator is straightforward for the forward part of the path. For the reverse path fragment, the created DFA must recognize the reverse of the original regular language. We rely on the property that regular languages are closed under the reverse operation, and the DFA accepting the reverse language can be created in an efficient way [114].

Growing of the frontiers in the bi-BFS and adding new entities to expand them is restricted by the appropriate DFA. Before an entity is placed on the frontier for the next expansion, a check is performed if the partial path leading to it is not rejected by the DFA. The entities creating partial paths that are rejected by the DFA are not added for the expansion, as the path they may create will not be accepted by the final DFA. When an intersection point is located, a final check of the joined path is performed by the forward DFA. If the forward DFA accepts a joined path, it becomes one of the located semantic associations that fulfills the provided regular path expression.

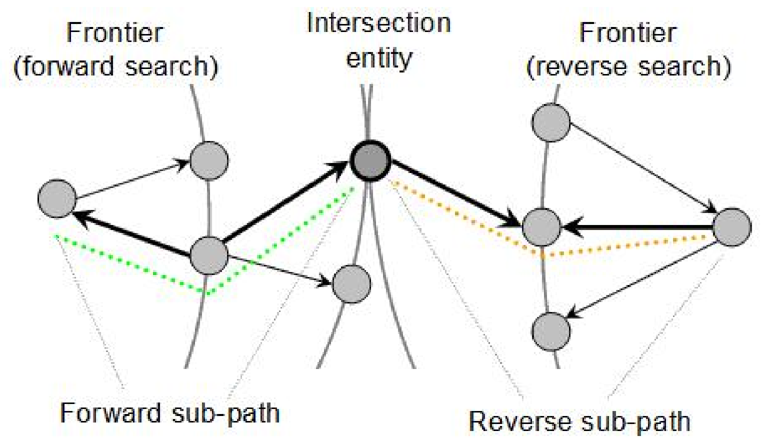


Figure 4.10 Forward and reverse sub-paths in bi-BFS search

Performing the path fragment check for each node before adding it to a next frontier for expansion causes the frontiers to grow slower than in case of searching for unrestricted semantic associations. Highly selective regular expressions can significantly reduce the growth rate of the frontiers, and as a result allow discovering of much longer semantic associations, as they do not exhaust the memory required for the search structures and limit the number of intersecting entities to check.

Single source path patterns cannot benefit from the speedup of bidirectional search. They are handled in a similar manner, with the use of only one (forward) DFA in conjunction with a standard breadth first search.

4.3.1. Proof of concept

One of our first functionality tests for semantic association discovery with the use of regular expressions over properties was to search for the N-Glycan biosynthesis metabolic pathway in the GlycO ontology [91]. GlycO (for "Glycomics Ontology") is a specialized ontology created for research in glycobiology. It contains knowledge that embodies semantically rich descriptions of carbohydrate structures, glycan binding relationships, glycan biosynthetic pathways, and the developmental biology of stem cells. The version used for our tests contained 573 classes and 113 types of named relationships. It has been semi-automatically populated with metabolic pathways and with more than 480 N-Glycans. GlycO, whose size is less than 15 Mbytes, is a relatively small ontology compared to the capabilities of the BRAHMS storage system.

Searching for the specific metabolic pathway presented in Figure 4.11 served as a proof of concept for semantic association search implemented in our SPARQLeR prototype.

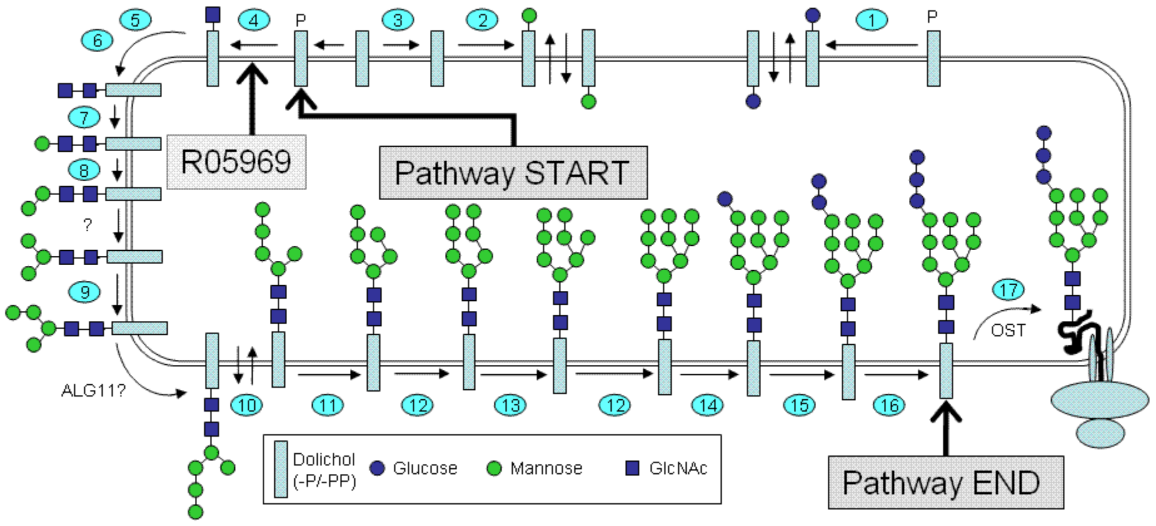


Figure 4.11 N-Glycan biosynthesis pathway with marked path query entpoints
(courtesy of Dr. Alison Vandersall-Nairn, University of Georgia)

A pathway is represented in GlycO by a sequence of biochemical reactions having defined their substrates and products. The reactions, as well as glycans, are first class objects in the ontology. We chose this pathway for its pattern of high regularity of relationships and a significant length. We searched for a fragment of the metabolic pathway that connects *dolichol phosphate*, goes through the reaction *R05969* (one of two possible at this step) and ends at *glyco* *peptide G00009*. The fragment consists of 15 chemical reactions, and as each reaction is modeled by an object with incoming and outgoing relationships, the length of this path reaches 30 relationships.

The prepared SPARQLeR query is presented below:

```
(19) SELECT list(%path) WHERE {
     glyco:dolichol_phosphate %path glyco:glyco_peptide_G00009 .
     %path  rdfs:member  enzyo:R05969
     FILTER ( length(%path) <= 30 &&
     regex(%path, "((-glyco:has_acceptor_substrate|
           -glyco:has_reactant) glyco:has_product)*" ) ) }
```

The resulting path was found almost instantly. This was due to the high selectivity of the regular path expression that allowed for expanding only relationships associated with chemical reactions, despite the significant length of the path. Additionally, a relatively small size of our test ontology allowed to limit the search space. This proof of concept test demonstrated the usefulness of the proposed SPARQL extension.

Note, that such long relationships are common in the biological domain. Their semantics is associated with metabolic pathways. However, in other domains, discovery of such long associations may be unnecessary. It is due to fact, that in many cases as the distance between resources increases, the clear meaning of a located association decreases and is hard to comprehend. For problems such as finding a conflict of interest presented in [13], long associations are not of interest, as they are virtually meaningless.

4.4. Scalability experiments for searching of regular paths

We have tested the scalability of our implementation of SPARQLeR and the use of regular path expressions for semantic association discovery using a collection of path queries against a modified subsets of the DBLP dataset [46]. We performed tests for discovering semantic associations between pairs of given endpoints, where the bi-BFS algorithm was used, and a single source discovery search executed by a standard BFS algorithm.

The tests were performed on machine with 2 Intel(R) Xeon(TM) 3.06GHz CPUs and 4Gb memory, running Red Hat Enterprise Linux 9.0. The C/C++ code was compiled using gcc (GCC) 3.2.3 20030502 (Red Hat Linux 3.2.3-56) with the '–O6' optimization flag.

4.4.1. Data Sets

For the scalability testing of semantic association discovery we used a modified version of the DBLP ontology generated from the data available in September, 2006. The originally created DBLP ontology contains the information about authors, published papers, year of publication, and other publication information starting from the year 1936. The full DBLP dataset contains 790,635 publications that have an assigned publication year. The cumulative number of publications included in the full DBLP ontology, starting from year 1936, is presented in Figure 4.12. We planned to search for long, meaningful associations that connect papers across different years by citations. Unfortunately, the ontology was not well populated with citations.

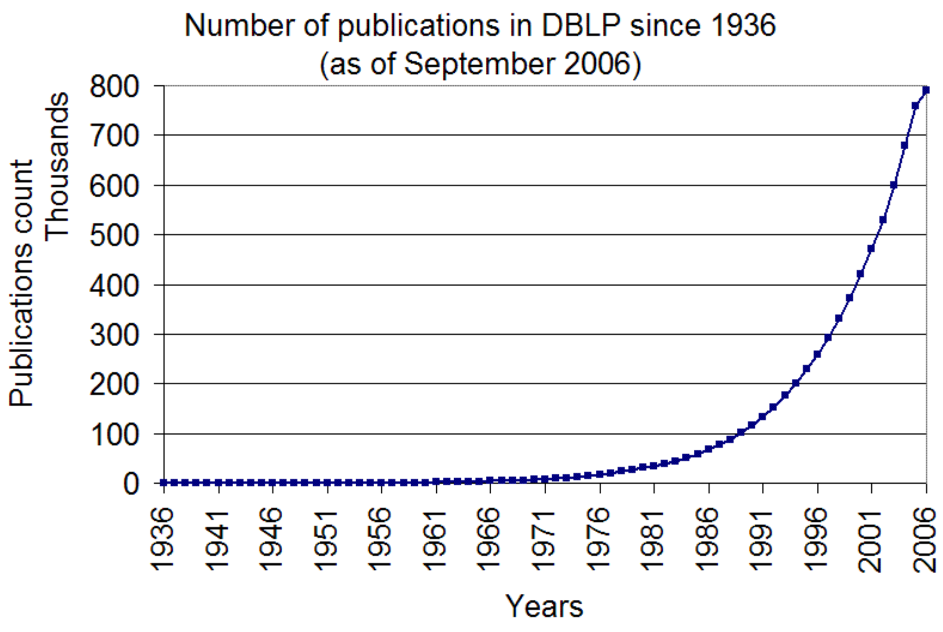


Figure 4.12 Cumulative number of publications in full DBLP ontology

Therefore, the citation records required for our semantic association search have been introduced artificially by adding random citations to each publication. For each publication, we synthetically created citation references to 1 to 10 randomly selected other papers, published before the selected one (determined by the publication year of both publications). We used a uniform distribution when assigning the number of citations to publications. The total number of randomly inserted citations in the full dataset reached almost 4.3 million.

The scalability of the semantic association discovery was tested using the same path queries and pairs of resources, while increasing the size of a dataset. For the performance tests we used a subset of DBLP that contained the publications in year 1981 and newer. It was due to the fact that the publication record before 1981 was very sparse, and almost all semantic associations that used the synthetically generated citations were ending on the same few publications. The full test dataset contained 760,369 publications and with the introduced citation records, in consisted of over 6.6M instance statements. The test datasets included publications from the increasingly wider time range. The smallest one consisted only of publications from year 2006, while the largest one included publications from year 1981 through year 2006. Using this method, 26 different test datasets were created. The sizes of these datasets (in number of instance statements) are presented in Figure 4.13.

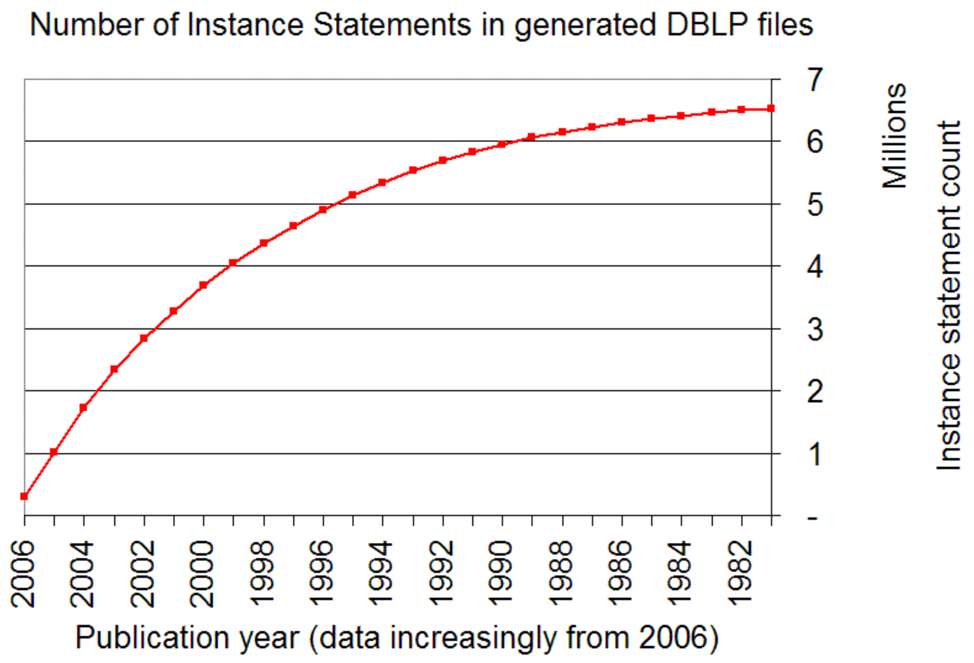


Figure 4.13 Modified DBLP test sets sizes

These datasets were used both for a single source association discovery and for association searched with two set entities.

4.4.2. Single source association exploration

We randomly chose 14 papers published in 2006 and executed single-source queries to find all paths leading to papers they cited, using the relation *opus:cites_publication*. Using the prepared datasets, the longest possible association was of length 26. A SPARQLeR query with a sample source publication entity in presented below:

```
(20) PREFIX dblp: <http://dblp.uni-trier.de/rec/bibtex/>
     PREFIX opus: <http://lsdis.cs.uga.edu/projects/semdis/opus>
     SELECT ?end_publication WHERE {
       <dblp:journals/ai/Huber06> %path ?end_publication
       FILTER ( length(%path)<=26 &&
         regex(%path, "(opus:cites_publication)*" ) ) }
```

The queries were performed on increasingly larger datasets, starting with articles published only in 2006 and ending with articles published during 1981-2006. Each query was executed 4 times against each dataset and average time was recorded.

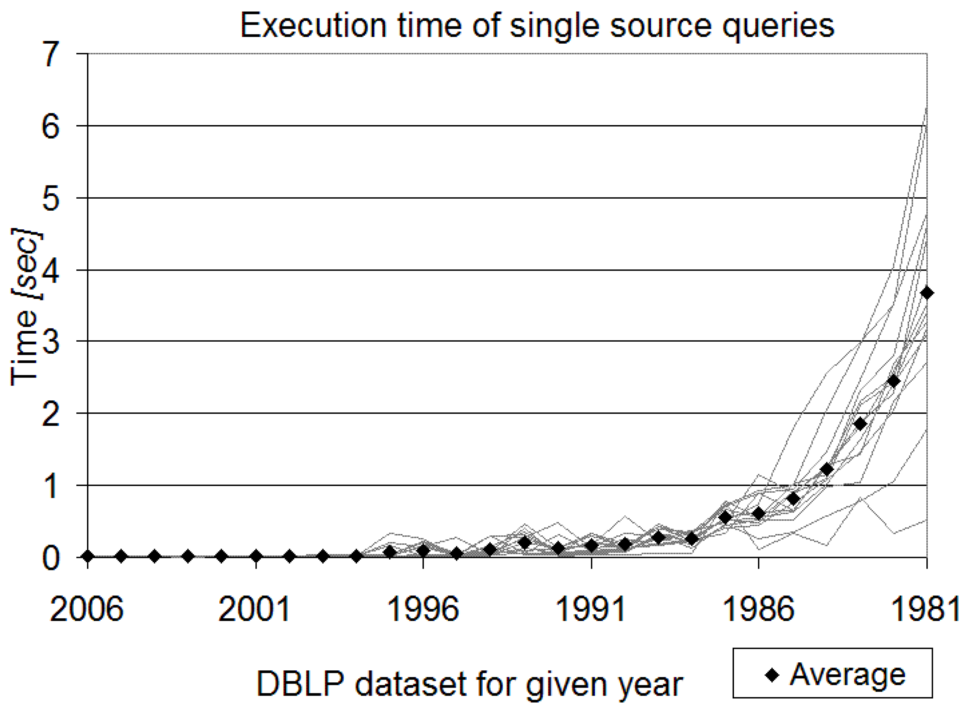


Figure 4.14 Execution times of single source queries in increasing DBLP datasets

In all cases, the execution time did not exceed 7 seconds, although for very long queries it was rising exponentially. We regard this time as acceptable for on-line applications. The details of execution times are presented in Figure 4.14. The number of located paths, presented in Figure 4.15, was increasing exponentially with the increase of the dataset.

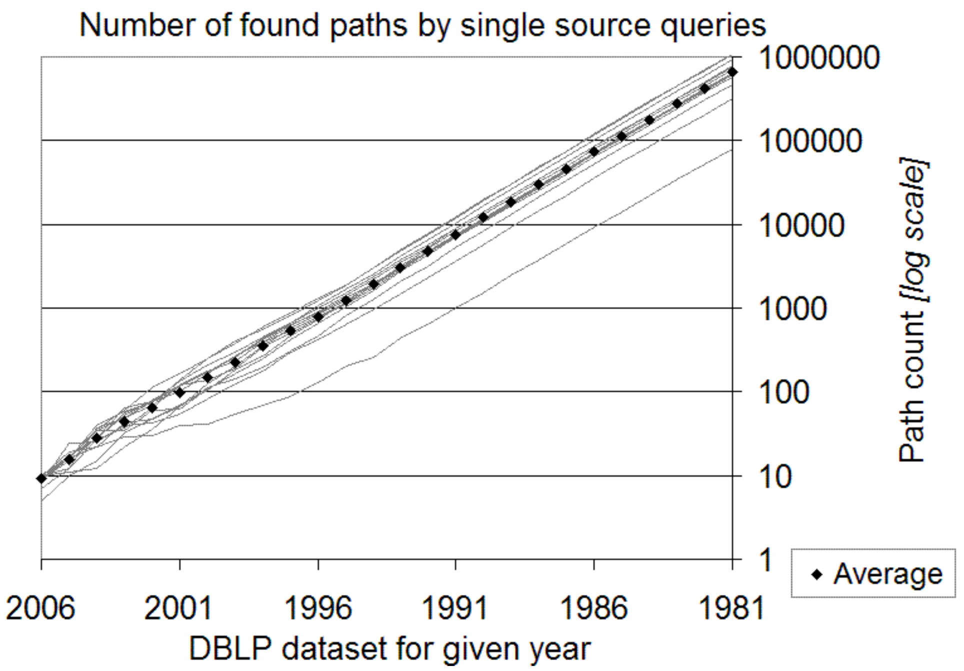


Figure 4.15 Number of located paths in single source queries (logarithmic scale)

For the largest dataset, each query located approximately 660,000 paths on average, which is still significantly lower than the theoretically possible number of 5^{26}, as implied by the method of synthetically adding citations.

4.4.3. Association discovery with two set endpoints

The second set of tests was performed on regular path queries that had both endpoints fixed. In the previous tests with single-source queries, we have identified 4 publications in relatively early years that were reachable by a relatively large number of paths from all of the previously chosen 14 starting publications. These 4 entities were selected as the endpoints for our path queries. A SPARQLeR query with two of the selected endpoints is presented below:

```
(21) PREFIX dblp:<http://dblp.uni-trier.de/rec/bibtex/>
     PREFIX opus:<http://lsdis.cs.uga.edu/projects/semdis/opus>
     SELECT list(%path) WHERE {
       <dbpl:journals/ai/Huber06>
           %path <dblp:conf/programm/BarbutiM80>
       FILTER ( length(%path)<=26 &&
           regex(%path, "(opus:cites_publication)*" ) ) }
```

As in the previous experiment, the queries were performed on increasingly larger datasets starting with articles published only in 2006 and ending with articles published during 1981-2006. Each query was executed 4 times against each dataset and the average time was recorded. This algorithm was implemented using a bi-BFS algorithms to locate the paths.

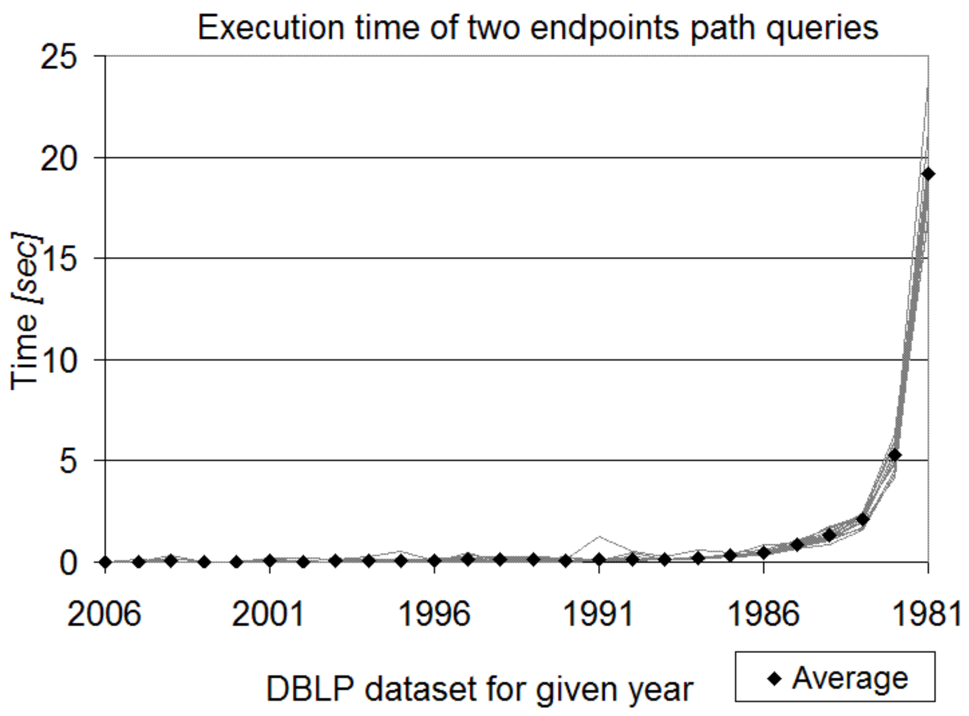


Figure 4.16 Execution times for double-source queries in increasing DBLP datasets

For smaller datasets and shorter paths, the execution time was negligible. The exponential growth was noticeable only for the largest datasets, however it did not exceed 25 seconds for any query. The detailed timing results are presented in Figure 4.16. In this experiment, the number of located paths, presented in Figure 4.17, was very small.

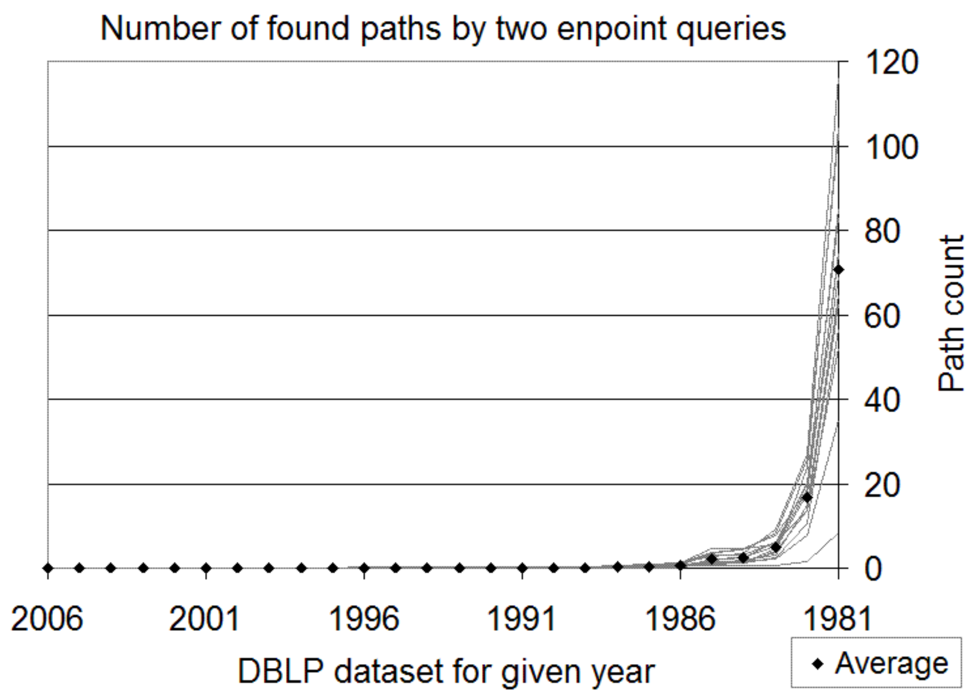


Figure 4.17 Number of located paths in double-source tests on increasing DBLP datasets

Despite the low number of the located paths, the execution time has been significantly higher than in a single-source test. It has been caused by the specificity of the datasets and the synthetically introduced citations. When following the citations from the newest papers to the older ones, the search space of the available papers decreases. However, the search space increases significantly when we follow the citations in the opposite direction.

The presented results prove the usability of the proposed SPARQL extension and its prototype implementation. We were able to locate regular paths of length up to 26 relationships in the ontology consisting of over 6.6M statements. With highly selective regular path expressions, the discovery of very long semantic associations, even in very large ontologies is possible, despite the exponential nature of the problem.

The SPARQLeR query language and its efficient implementation using the BRAHMS ontology storage system allowed us to successfully perform experiments with the ontology-based text categorizations described in the next chapter. Support for capturing both structural and semantic requirements in the query was very important in performing the tasks required by the ontological categorization.

CHAPTER 5 ONTOLOGY-BASED TRAINING-LESS TEXT CATEGORIZATION: OMNICAT

In this chapter we present a novel method for the automatic text classification, OmniCat [53] [54] [55]. The method relies on the domain knowledge represented in the form of an ontology to perform the categorization task. More specifically, it concentrates on the recognized named entities and relationships in the document text to measure the semantic similarity of the created thematic graph to the categories, defined as ontology fragments, and perform the categorization. In the proposed text categorization method, the ontology effectively becomes the classifier. OmniCat does not require a training set of pre-classified documents to learn about the distinguishing features of categories. Instead, it allows defining categories as fragments of ontological knowledge. In this chapter we present formalisms for dynamic definition of categories as ontological contexts of interests. With the change of interest, documents can be instantly reclassified without a need of retraining of the classifier or supplying a modified training set, which is one of the main strengths of the proposed method. In the forthcoming sections we describe details of the proposed method and required steps to transform a document text into a thematic graph. Finally, we propose two approaches to categorization of the created thematic graph using ontology: (1) a bottom-up category discovery, and (2) a top-down projection of the defined categorization contexts.

5.1. Ontology in automatic text categorization

Text categorization is a task of assigning one or more of the previously defined categories to an analyzed document, based on the document's content. Research on the automated text classification, which started in the 60s, is still a very active area of investigation, and multiple methods and approaches have been developed. Majority of classical text categorization methods create the categorizer by learning the distinguishing features of the category from examples of previously categorized documents. Later, they utilize this knowledge to assign categories to previously unseen documents. A short review of the current text categorization methods is presented in Section 2.4, whereas a more comprehensive survey has been presented in [96].

When people classify text documents, they rely both on (1) the definitions of categories, usually expressed in the natural language form and their general knowledge of the classification domain, and (2) specific descriptions, such as concepts, terms, phrases, names, events, and others that allow deciding whether a document belongs to this category. Definition of a category in the natural language can be expressed quite precisely and with a high level of detail, or provided by means of general terms and commonly used associations.

The decisive factors that help to decide if a document belongs to a specific category are either stated explicitly in the category definition or provided implicitly, referring to the person's factual knowledge about the domain. Using our knowledge about the surrounding world we are able to identify several entities in the text, such as people, companies, places, and other. We can recognize the roles of the recognized entities, their types, and also we are able to establish meaningful associations among them. Additionally, using the general knowledge and contextual information, we can infer the information that is not explicitly present in the document, but may significantly improve its understanding.

In the presented automatic text categorization method, OmniCat, we propose to utilize the analogous approach and apply domain knowledge for text categorization. We argue that named entities recognized in a text document, together with meaningful associations connecting them, can determine the categorization of the document. Appropriate structures and semantics can be provided by a domain ontology. An ontology is one of the knowledge representation formalisms understandable to people, which is also machine readable. In computing, ontology is “a representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain.”[1]

A good quality ontology with appropriate entity labels can be successfully used for entity recognition in a natural language text. The associations between entities may have two independent sources. They are already present in the ontology or they can be extracted directly from the document using NLP techniques. Associations taken directly from a document comprise factual, direct knowledge present in the document. Relationships induced by the ontology provide additional, domain specific knowledge about entities that may not be included in the document, but can serve as important contextual information for categorization.

Knowledge in the ontology is represented by entities’ associations and classification of the entities into a taxonomy. Closely related entities represent knowledge about one or a few similar domains. It is also common that closely related entities are classified into the same taxonomy tree. Entities from different domains are related by longer associations or, even when we analyze only a specific ontology fragment, they can even be unrelated. We use the assumption of close relatedness of entities from similar domains in determining the dominant subject of the document, and finally, its categorization.

Classification categories in OmniCat are defined as dynamic ontology contexts. They include classes, instances, relationships, and combination of the above that may be used to

---

[1] http://en.wikipedia.org/Ontology (information science)

identify specific semantic structures. In defining a category, we may differentiate the importance of various elements in order to shift the focus of categorization towards them.

In the proposed approach, the ontology effectively becomes the classifier. Analyzed text, by recognizing and disambiguating entities, extracting and imposing known relationships, is transformed into a semantic graph. As a result, we classify a semantic graph to dynamically defined ontology contexts. Therefore, the quality of the used ontology determines the quality of categorization. To be able to achieve meaningful results we require that the ontology:

- cover the categorized domain(s),
- include a rich instance base of named entities and relationships between them,
- provide proper labels for named entities that enable their recognition in the unstructured text,
- have entities categorized according to a class taxonomy included in the ontology.

Richness of instances and labels describing individual entities is essential for properly recognizing a maximum number of named entities in the document text. It is the first and most important task in transforming the document text into a thematic graph. Domains covered by the ontology characterize the scope, in which an ontology can be effectively used for categorization. Variety of domains provide more flexibility and widens the application of ontology for the purpose of text categorization. Finally, the quality of the used taxonomy and the categorization of individual entities directly influences the quality of the performed categorization, as categorization of the whole document is a result of the individual categorizations.

We only can list the above requirements as guidelines for choosing an appropriate ontology for the text categorization, as they cannot be expressed using the current ontology quality metrics.

5.1.1. The Wikipedia ontology

Wikipedia is a free, multilingual, open content encyclopedia. It is considered to be the world's largest encyclopedia. The number of articles in the English version exceeds 2,000,000 and is still growing. They cover a multitude of domains of basic, encyclopedic knowledge, in many cases even describing highly specialized knowledge. Majority of articles are classified according to a categorization scheme, which is organized more as a thesaurus rather than a taxonomy [119]. Multiple classification, which is not uncommon in Wikipedia, allows looking at the same articles in multiple different contexts.

This broad and detailed encyclopedic knowledge can be automatically transformed from Wiki representation into an ontology. Tools, such as described in [17], allow to convert Wikipedia into an RDF/S ontology, and treat each entry as a named entity. In our opinion, the Wikipedia ontology fulfills most of the presented requirements for suitable ontology for text categorization purposes.

The major advantages of the Wikipedia ontology include the richness of represented domains, high number of entities, and a elaborate categorization scheme. A large number of entities (web entries), together with multiple alternative names for each entity, makes it a perfect resource for recognizing a wide variety of entities in the unstructured text. Extensive links connecting entity descriptions using *infoboxes*, *templates*, and simple hypertext references (*hrefs*) define a simple, yet strong backbone of semantic associations between entities. Furthermore, a broad category graph provides a usable categorization scheme. The provided structure and multiple classification of entities facilitates discovery of related entities and helps the categorization algorithm to focus on a specific thematic interpretation of the analyzed graph.

Finally, continuous updates and additions of new entries make Wikipedia a perfect source of precise and up-to-date encyclopedic knowledge.

However, the Wikipedia ontology, directly after a conversion to its RDF/S representation, cannot be successfully applied for text categorization. There are a few issues, associated with the web-based origins of Wikipedia that have to be modified before it becomes a suitable ontology for entity recognition and text categorization. These include extending the available name aliases for entities, applying common prefixes to names of relationships present in infoboxes and templates, and special handling for included page templates that contain other named entities.

Entities in the Wikipedia ontology are uniquely identified by the corresponding entry names or redirections. In case of ambiguous names, Wikipedia uses a convention of adding, enclosed within parentheses, contextual information to an entry name. Although this method is very useful for identifying entities, a direct use of such provided labels for recognizing entities in text will not work correctly. In reality, a document does not contain direct contextual identification of entities. For example, looking for a Jaguar vehicle, one is highly unlikely to find a "*Jaguar (car)*" phrase in the document, which uniquely identifies the required entity in Wikipedia. The string "*Jaguar*" points to a well-known animal in *felidae* family, which although is a generally correct association, does not satisfy the automotive context. A sub-set of labels and entities that can be associated with "Jaguar" in Wikipedia is presented in Figure 5.1.

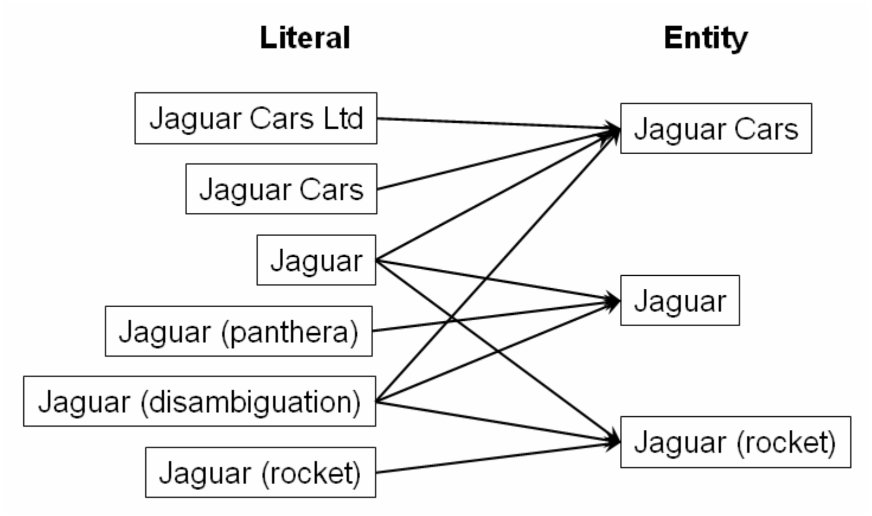
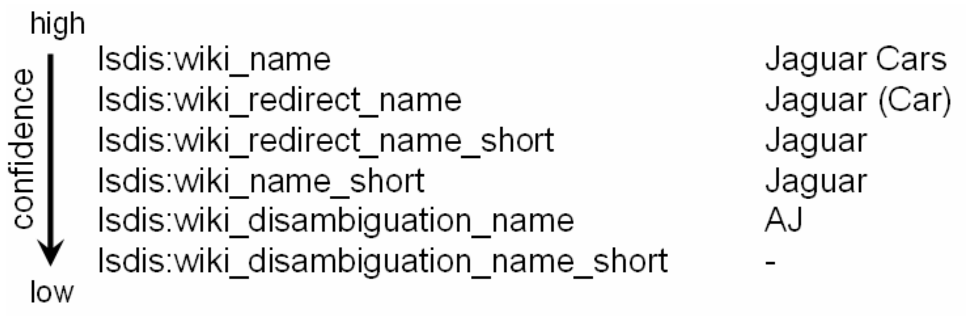


Figure 5.1 Names of ambiguous entities

To overcome the problem of ambiguous names and to enable a more correct entity matching in the document, we added specific relationships connecting ambiguous or context-less labels with the appropriate entities. The introduced relationships represent the level of confidence in how uniquely a label describes an entity. The introduced relationships and their example use for Jaguar are presented in Figure 5.2.

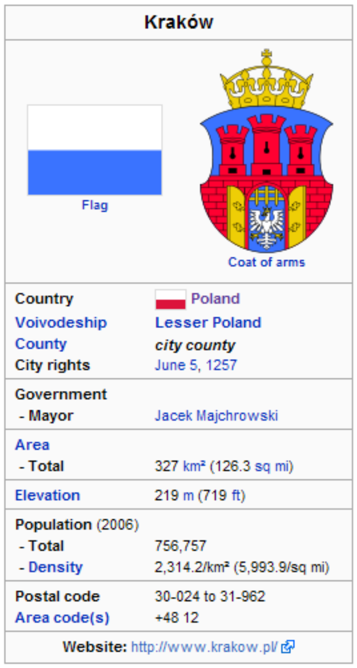

high
confidence
low

lsdis:wiki_name	Jaguar Cars
lsdis:wiki_redirect_name	Jaguar (Car)
lsdis:wiki_redirect_name_short	Jaguar
lsdis:wiki_name_short	Jaguar
lsdis:wiki_disambiguation_name	AJ
lsdis:wiki_disambiguation_name_short	-

Figure 5.2 Introduced relationships to match labels with entities in Wikipedia

Wikipedia consists of a network of related web pages, which are connected by hyperlinks. They define only implicit semantics of relationships the connected entities. Some pages, however, contain more structural information in the form of *templates* or *infoboxes*. The example Wikipedia code for the infobox including information about the city of Krakow, Poland, and its rendering on the webpage is presented in Figure 5.3.

```
{{Infobox Settlement
  name = Kraków
  image_flag = Flag of Krakow.svg
  image_shield = POL Kraków COA.svg
  subdivision_type = Country
  subdivision_name = {{POL}}
  subdivision_type1 =
    [[Voivodeships of Poland
      |Voivodeship]]
  subdivision_name1 =
    [[Lesser Poland Voivodeship
      |Lesser Poland]]
  subdivision_type2 =
    [[Powiat|County]]
  subdivision_name2 = ''city county''
  leader_title = Mayor
  leader_name = [[Jacek Majchrowski]]
  established_title = City rights
  established_date =
    [[June 5]], [[1257]]
  area_total_km2 = 327
  population_as_of = 2006
  population_total = 756757
  population_density_km2 = auto
  elevation_m = 219
  postal_code_type = Postal code
  postal_code = 30-024 to 31-962
  area_code = +48 12
  website = http://www.krakow.pl/
}}
```



Kraków

Country	Poland
Voivodeship	Lesser Poland
County	*city county*
City rights	June 5, 1257
Government	
- Mayor	Jacek Majchrowski
Area	
- Total	327 km² (126.3 sq mi)
Elevation	219 m (719 ft)
Population (2006)	
- Total	756,757
- Density	2,314.2/km² (5,993.9/sq mi)
Postal code	30-024 to 31-962
Area code(s)	+48 12

Website: http://www.krakow.pl/

Figure 5.3 Example of Wikipedia infobox entry and its rendering in the webpage

The information encoded in infoboxes and templates provides a deeper semantic insight into the relationships between the connected entities. Using a special handling of such page fragments, the named relationships can be added to the ontology. However, the interpretation of such relationships during text categorization is rather limited, as the understanding of their meaning and importance would require a significant and complex NLP analysis, which is beyond the scope of this dissertation. We can only assume that relationships defined by the infoboxes and templates are much more important than the normal hyperlinks, and that the infoboxes carry the information of better semantic quality than the templates.

The last correction to the original DBpedia conversion algorithm is in handling of templates included in the pages. Originally, the algorithm created a separate blank node that represented the template and linked the entity to it. Although this approach is semantically correct, as the template can be treated as a separate object, it also separates the entities by one additional relationship, unnecessarily lengthening the association between them. For shortening the associations between the page entry and the entities listed in a Wikipedia template, we decided to omit the blank node and link the entities in the template to the main page entity using direct relationships. In this solution, they remain closely related, as it was initially intended by using the template.

The Wikipedia ontology, with the introduced modifications can be applied for automatic text categorization. Before formally describing the categorization algorithm details, we present an example of the categorization process using this ontology.

5.1.2. Example of a text categorization with Wikipedia

Let us present a fragment of a recent news article to illustrate the process of ontology-based categorization. We underlined the words and phrases that were matched to entity names in the Wikipedia. Bold words and phrases represent the entities selected for text categorization.

> United States **presidential candidate** **Barack Obama** **today** gave a **speech** concerning **racial** division, which has become a major issue in his **campaign** after his former **pastor**'s racially-charged statements were widely reported in **the media** last week.
>
> The 45-minute speech, which **Obama**'s advisers say was written by **Barack** himself, was delivered at **the National Constitution Center**, a **history museum** in Philadelphia, Pennsylvania. It was broadcast live on national **television**.
>
> **Obama** said Rev. **Jeremiah Wright**'s statements “expressed a profoundly distorted view of this country - a view that sees **white racism** as endemic,

> and that elevates what is wrong with **America** above all that we know is right with **America**.” In one of Wright's more controversial **sermons**, he said **blacks** should damn **America** for mistreating and ignoring them. […]
>
> **Hillary Clinton**, also campaigning in Philadelphia, gave a positive **comment** about **Barack**'s speech. “Issues of **race** and **gender** in America have been complicated throughout our **history**, and they are complicated in this primary **campaign**,” said **Clinton**. “There have been detours and pitfalls along the way, but we should remember that this is a historic moment for the **Democratic Party** and for our country. We will be nominating the first African-American or **woman** for the **presidency of the United States**, and that is something that all Americans can and should celebrate.” […]

The proposed ontology-based categorization relies on entities and relationships among them. It requires a transformation of an unstructured document text into a semantic graph. First, we match phrases in the text (underlined) and associate them with the entities in the ontology. Note, that some of the phrases can be associated with multiple entities and thus will require further disambiguation. Race is one of the examples of ambiguous phrases. According to Wikipedia, it can be matched to Racing, Race (biology), Race (classification of human beings) or even to several people whose last name is Race.

On the basis of the Wikipedia ontology, we can establish a number of relationships among the recognized entities, such as that Barrack Obama and Hilary Clinton are both presidential candidates and they both represent the Democratic Party. Knowing the phrases that represent entities, NLP parsing of the sentences can add relationships that may not be present in the ontology. The discovered relationships and the relationships induced by the ontology enable us to create a semantic graph that represents the content of the analyzed document.

The created graph can be composed of multiple disjoint connected components or just a few cohesive elements connected by a few relationships. Each component can be associated with a different domain, as we previously assumed that closely related entities in the ontology should be classified into the same or similar domains. The component that presents the best fit to the ontology and offers the largest coverage of a topic should be selected as the dominant thematic graph. This dominant graph represents the main topic of the document and should be used for establishing the categorization of the document. The selection of the dominant thematic graph should effectively disambiguate any entities that were incorrectly matched in the first step. If a single phrase matches multiple entities, in most cases they belong to different contexts. Choosing a specific interpretation by selecting the dominant thematic graph sets a proper context and eliminates the entities not related to the main topic of the document. In the example above, Race would be finally matched to Race (classification of human beings). The dominant thematic graph created for the above example is presented in Figure 5.4.

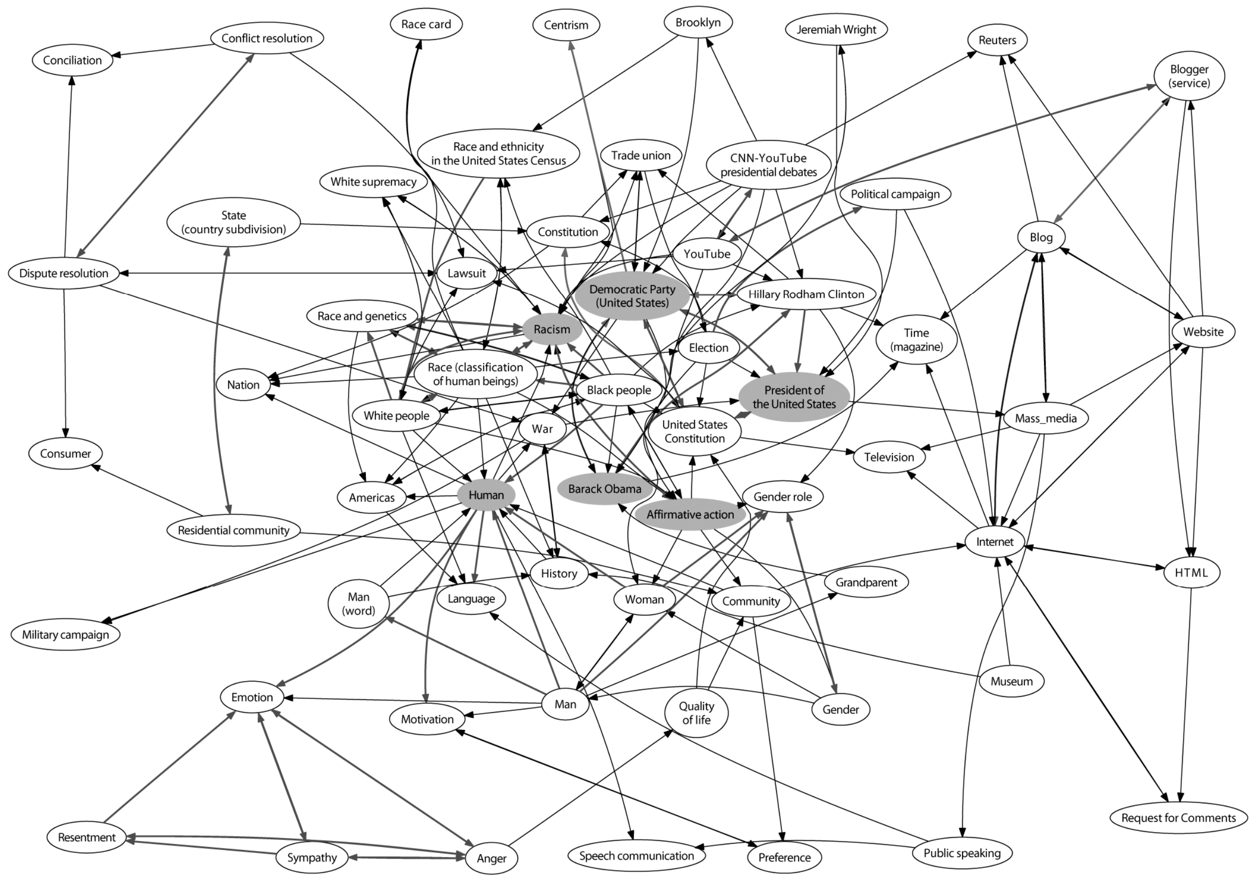


Figure 5.4 Dominant thematic graph created from the example article

Ideally, the selected thematic graph covers one specific domain. In reality, it may cover multiple domains, offering different interpretations of topics covered by the document that depend on the selection of entities included in the analysis. In the graph presented in Figure 5.4, we can distinguish a highly connected central part of the graph and a number of related, but peripheral entities. To narrow the number of possible interpretations, the focus of the categorization should be concentrated on the highly connected core of the graph.

The entities that belong to the core should be either most central to the graph, in terms of the geographic centrality measure, or the most important ones. We believe that the best hub and authoritative entities, discovered by the HITS algorithm [62] are of high importance. For the example article, the entities selected as the core of the graph categorization are presented as shaded in Figure 5.4.

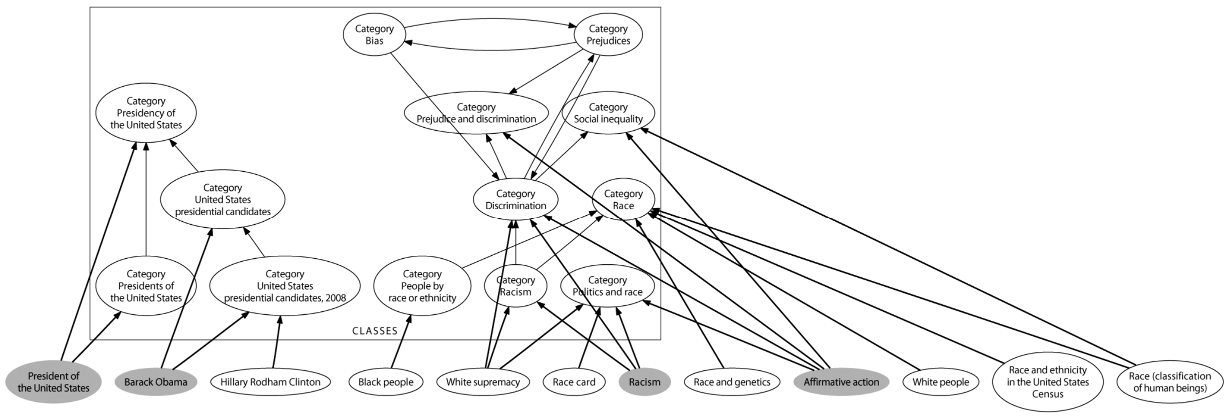

Category Bias
Category Prejudices
Category Presidency of the United States
Category Prejudice and discrimination
Category Social inequality
Category United States presidential candidates
Category Discrimination
Category Race
Category Presidents of the United States
Category United States presidential candidates, 2008
Category People by race or ethnicity
Category Racism
Category Politics and race
CLASSES
President of the United States
Barack Obama
Hillary Rodham Clinton
Black people
White supremacy
Race card
Racism
Race and genetics
Affirmative action
White people
Race and ethnicity in the United States Census
Race (classification of human beings)

Figure 5.5 Fragment of the recognized category hierarchy for the selected entities in the graph

The categorization of the dominant thematic graph should be performed starting from the graph core. Our categorization algorithm could assign a number of the most likely Wikipedia categories that cover the graph core and have a strong support in satellite entities included in the thematic graph. Figure 5.5 illustrates a small fragment of the Wikipedia category hierarchy for the selected entities from the example graph. The top selected categories for the example graph are presented below:

- Democratic_Party_(United_States)
- Society
- Political_parties_in_the_United_States_by_ideology
- Politics
- Race
- Presidency_of_the_United_States

5.2. Ontology-based categorization algorithm

Our training-less ontology-based categorization method is founded on establishing the level of a semantic similarity between the thematic graph that represents a document, and a category provided as relevant ontology fragments. The algorithm consists of three main steps: (1) the construction of a semantic graph that contains all possible recognized entities and domains from the original document text, (2) the selection of a thematic graph that effectively sets the major domains for further analysis and categorization, and (3) the categorization of a chosen thematic graph into the domain taxonomy using one of the proposed methods. The outline of the categorization algorithm is presented below:

Semantic graph construction

- Named entity identification – phrases describing the entities (entity labels) in the ontology are matched in the text; for each spotted phrase, all associated entities are added as nodes to the created semantic graph with an initial weight based on the confidence of the match; relationships used for relating literals with entities and their confidence levels are presented in Figure 5.2.
- Entity relations extraction – edges between the recognized entities are created based on the NLP analysis of sentences in the document (we used the chunk parser presented in [113]); this step extracts the relations between the spotted phrases directly from the analyzed document.
- Connectivity inducement – additional edges between the nodes in the semantic graph are created based on the relationships existing in the ontology and connecting the entities corresponding to the nodes; each edge is assigned a weight based on the importance of the relationship in the ontology schema; for the Wikipedia ontology we

use infobox relationships, templates, and hypertext references for properties induced in the graph.

Dominant thematic graph selection

- (optional) Removal of special type entities – entities identified as places, dates or time-related are removed from the semantic graph; this step is specific to categorizing news documents as entities related to spatio-temporal domain tend to become central entities, and drive categorization into unwanted domains.
- Information propagation – the node weights are propagated to their neighbors in order to establish the most authoritative entities in the graph.
- Connected component identification – the connected components in the semantic graph are identified, treating the graph as undirected.
- Dominant thematic graph identification – the largest and most important connected component of the semantic graph is selected as the dominant thematic graph.
- Core selection – the most important and/or central entities in the thematic graph are identified; they form the core of the thematic graph.

Categorization of a dominant thematic graph

- Bottom-up classification – discover taxonomy classes that (i) offer the best coverage of the dominant thematic graph, (ii) include the most of the authoritative and central entities, (iii) are closest to the original classification of graph entities.
- Top-down classification - project the *categorization contexts*, described in more detail in Section 5.4, onto the graph to find category that provides the best fit for the dominant thematic graph.

Both categorization approaches are described in detail in subsequent sections.

5.2.1. Semantic graph construction

The first step in preparing the document text for ontology-based classification is the construction of the semantic graph, based on the text of the document. It starts with the identification of known named entities in the document. The ontology entities occurring in the analyzed document are identified by matching document phrases with entity literals. Entities in the ontology have one or more literal properties associated with each of them. We assume that these properties define the entity name (usually known as its label), and may also specify the entity name's synonyms (aliases). For each property type that associate literal with the entity, as presented in Figure 5.2, we assign the weight of an entity match. The weight reflects our level of confidence in how well and unique a given label describes the entity.

Word stemming and stop words removal may be applied to the document text before the entity identification step. Modifications of a document text may cause that an exact match of a label in the ontology and a phrase in the document may not be possible. To accommodate for such cases we allow partial matches, giving preference to an exact or most complete match to the entity's label.

An entity can be matched in multiple places of the document. It is an important information, which is analogous to the term frequency used in the traditional text categorization methods. It suggests a higher importance of the entity in the document, and should be reflected by the increase of the initial weight assigned to the entity. However, the importance cannot be increased linearly with the number of occurrences, as it would create disproportions between matched entities so significant, that the categorization process would be biased by only few frequent entities.

To accommodate the confidence level in the entity match, completeness of the match and the multi-occurrence of an identification phrase, we use the following formula (1) to assign an initial weight of the entity.

$$w = 1 - \frac{1}{1 + \sum_{i=1..n} p_i * s(l_i, mp_i)} \qquad (1)$$

For each entity, we sum the products of confidence of the relation used for entity description (p_i) and the similarity s of the matched phrase (mp_i) and the associated literal (l_i). The function s measures the completeness of a match and similarity between the matched phrase mp_i and the original literal l_i in the ontology, with respect to the removed stop words and/or stemming. When stemming and stop words removal is not used, the similarity function s degenerates to simple string matching.

The number of appearances of entity phrases in the document and our confidence in phrases' matches is reflected by the sum in the equation. We utilized the property of $1 - {}^{1}/_{n}$ formula to incorporate the importance of multiple appearances of entity matches in the document, while preventing significant disproportions of initial weights assigned to matched entities. We found the proposed formula useful for assigning the initial weights for matched entities. There are other alternatives, as the ones proposed by the authors in [126], that can be also applied to indicate the initial importance of a matched entity in the graph.

A literal in the ontology may be associated with multiple entities, since different entities may share names or aliases. Therefore, the number of initially identified entities may be higher than the number of matched phrases in the text, as incorrectly identified entities (false positives) may be included in it. We create a semantic graph using all of the identified entities. The disambiguation and elimination of incorrectly matched entities is performed in further steps of the algorithm.

The phase of relationship identification directly from the text utilizes the previously matched phrases in the document. We parse individual sentences of the document in order to find any additional direct associations between the entities identified by the matched phrases. This approach uses the parse tree to find whether phrases (entities) are syntactically connected. We found such information as sufficient for establishing a simplified, yet important association between entities. We do not perform the relationship extraction or the analysis of the semantics and importance of the discovered associations, as complexity of the process is beyond the scope this dissertation and the proposed text categorization method.

The final phase of forming a semantic graph is inducing relationships between the recognized entities. We use the knowledge from the ontology to add known associations among the identified entities to the formed semantic graph. This can be seen as adding the background knowledge to semantically relate the matched entities using the information that is not present in the document. Each added property type from the ontology has an assigned importance factor, indicating the strength of the relationship between the entities. For the Wikipedia ontology, most important are the relationships created by infobox connections, followed by templates, and finally hypertext references, as least important. The induced ontological knowledge offers plausible semantic interpretations for the co-occurrence of the entities and provides important information in determining the document classification.

5.2.2. Thematic graph selection

The previously created semantic graph contains a mix of knowledge extracted from the document and imposed by the ontology. It may contain incorrectly recognized entities and cover multiple domains. This step of the algorithm is similar to the feature selection in the traditional text categorization approaches, and focuses the thematic graph on a few dominant topics of the document.

In the first step of the transformation of the semantic graph, we remove the recognized entities related to spatial or temporal domains in the ontology. It is implied by a specific nature of the news documents and semantic graph originally created from them. Typically, news articles contain the information about the place (where), the date/time (when), and the content (what), which are in most cases orthogonal to each other. However, in the created graphs, the entities that identify places and dates often become important or central ones. They conceal the important entities of the "real content" from the categorization algorithm and skew the categorization results.

Our categorization algorithm focuses on the most important entities from the analyzed document. Such entities not necessarily have to be the most frequent ones in the document or the ones with the best match. We treat the semantic graph as a weighted directed graph and apply the HITS algorithm [62] for weight propagation. In fact, the algorithm uses the semantics encoded in the assigned weights and the structural properties defined by the relationships with their importance to identify the semantically important entities. Later the identified hubs and authorities become the core entities in the graph. The propagated weights represent the importance of the entities in the graph.

Narrowing the focus to one or a few domains requires selecting the representative thematic subgraph. The selection of the thematic graph is based on the assumption that the entities within one domain are closely related to each other and form a connected component in the semantic graph. The component with the largest number of entities and the highest total of entity weights is selected as the dominant thematic graph for the document. In a rare case, it is possible that few components have a similar size or weight. In this case, we conclude that the original document describes multiple domains with similar importance, and therefore the dominant thematic graph should consist of all these subgraphs.

The selection of the dominant thematic graph sets an interpretation context and effectively disambiguates many of the incorrectly matched entities. Furthermore, selecting only specific graph components for the categorization in the dominant thematic graph, removes the entities that are not related or are weakly related to the major domain represented in the document.

The last step in preparing the dominant thematic graph for the categorization is locating its core entities that become the most important ones in categorization. Naturally, the top hubs and authorities are included in the graph core. Additionally, we search for the central entities in the dominant thematic graph, as they can be treated as *topic landmarks*. We use the geographical centrality measure (2) to locate most central entities [90].

$$Centrality(v_i) = \frac{1}{\sum_j d(v_i, v_j)} \tag{2}$$

The $d(v_i, v_j)$ represents the shortest path distance between vertices v_i and v_j in the undirected, not weighted graph.

As a result, *the dominant thematic graph* represents the main focus of the document, and the *core of the thematic graph* includes the top authoritative and the most central entities.

5.3. Direct ontology graph categorization

The first proposed method to categorize the thematic graph is a bottom-up approach. We rely only on the ontology and the classification of the individual weighted entities in the graph to a given taxonomy. On top of the recognized entities, we build structures of class hierarchies and search for the classes that best describe a significant portion of the graph.

The previously created dominant thematic graph includes the entities and relationships present in the ontology. In this sense, it is semantically similar to some ontology fragments. Finding the best categories that describe the dominant thematic graph shifts the attention from the instances (entities) to the ontology class taxonomy. The selected classes in the taxonomy should represent the best possible fit to the largest part of the defined ontology fragments. It requires an optimization of a number of different, possibly conflicting objectives. The best category should:

- maximize the entity coverage (be a class or super-class of) of entities in the thematic graph,
- provide the highest total weight of the covered entities,
- be very close (in the hierarchical distance) to the covered entities and, if possible, be the most specific (lowest) class in the taxonomical hierarchy,
- closely cover the highest number of the core entities (at least one) in the thematic graph.

Taking into consideration the properties of the best coverage class, we use the following formula (3) for calculating the class score:

$$(3) \qquad s_{Ci}(h_{\max}) = 1 - (1 - \frac{1}{1 + \sum_j \frac{w_j}{h(C_i, e_j)^2} + \sum_k \frac{w_k}{h(C_i, e_{Ck})}})$$

where s_{Ci} is the score for class C_i, that includes reachable entities e up to depth h_{max} ; e_j and e_{Ck} represent respectively an entity and a core entity reachable up to depth h_{max} from class C_i ; w_j and w_{Ck} are weights of entity e_j and core entity e_{Ck} ; $h(C_i, e)$ is the hierarchical distance between category C_i and the covered entity e. The first summation is over all of the entities reachable from category C_i up to depth h_{max}, while the second one includes only the core entities. Any class which does not cover at least one of the core entities is assigned a zero score and not considered to represent the main topic of the document.

In computation of the class score we utilize the same formula as the one used for assigning initial entity weight (1). To avoid domination of classes with high number of low-weighted covered instances, we placed the sums in the denominator. The weights of the individual entities are adjusted by a hierarchical distance h from an entity to the class to capture the relevance of the class for the entity. To favor entities in the graph core in categorization, their weights are adjusted by a smaller factor than the weights of all remaining entities. Again, this formula has been created to accommodate the properties we defined for the best coverage class. Other solutions for the class score formula are possible and valid, and their details depend on defining what is in the center of interest for the categorization.

Classes with positive score values are ranked according to their categorization score and represent the categorization of the whole document, relatively to the taxonomy included in the ontology schema. At this step, the document has assigned multiple ranked categories that describe its content based only on the ontological knowledge.

Classification to externally defined categories requires matching between the user-defined categories and ontology classes. In the simplest case, assigning an external category is based on finding an intersection between the highest ranked classes and category definitions provided as sets of classes. The decisive factor for categorization may depend on the total score or the number of the classes in the intersection.

In the Wikipedia case, where the category hierarchy forms a thesaurus, the scoring function should strictly limit the maximum hierarchical distance of the included categories from the thematic graph. It is due to the characteristics of a thesaurus, that with the increase of the hierarchical distance, the relevance of the assigned category decreases. In the experiments we estimated the maximum distance that retains the relevance to be between 4 and 5 relationships, and we used these numbers in the proposed class scoring formula.

The proposed class score function favors the classes that are closest to the thematic graph and do not include the information about their depth in the ontology. In case when a user is interested in classifying into the most specific classes in the taxonomy, the scoring function should be modified to accommodate it. The proposed scoring functions present the intuitive approach for assigning appropriate scores based on the defined functional requirements imposed on the method. They are also the ones we used in our experiments. In the future, we intend to experiment with other formulas, more specifically with the ones that include statistical information from the ontology.

5.4. Categorization into contexts

Another method for classification of the dominant thematic graph relies of the dynamic specification of classification categories by ontological contexts. We present our notion of an ontology context, defined as a projection over the ontology classes and instances (entities), and how the ontology contexts can be used as classification categories.

5.4.1. Context as an ontology sub-graph

We specify a *categorization context* as an ontology subgraph or an ontology fragment. Our definition of a categorization context is to some extent based on prior works on the notion of views in semi-structured databases [9] and in ontologies [30]. Previous research on contexts and their formal specification can be found in [37] and [72].

In the following definitions, *R* represents an RDF description base, while *S* an associated RDFS schema.

Def. 5.1 The *hierarchical distance* between an instance entity *e* from a description base *R* and a class *c* from an RDFS schema *S*, denoted as *distH(e,c)*, is defined as the length of the shortest path formed by one *rdf:type* and zero or more *rdfs:subClassOf* properties connecting *e* and *c*. In case the entity *e* is not an instance of class *c* (directly or via the *rdfs:subClassOf* properties), *distH(e,c)* is set to 0. By extension, the hierarchical distance between an instance entity *e* and a set of classes *C*, denoted as *distH(e,C)*, is defined as the minimum, positive value of all *distH(e,c)*, where $c \in C$. If *e* is not an instance of any of the classes in *C* (directly or via the *rdfs:subClassOf* properties), *distH(e,C)* is set to 0. □

Def. 5.2 Let *C* be a set of schema classes included in an RDFS schema *S*. A *projection* of classes *C* onto an RDF description base *R* is a set of instance entities in *R* together with their associated hierarchical distance to *C*, defined as:

(4) $$\Pi(C,R) = \{\, e(k) : e \in R \wedge k = distH(e,C) \wedge k > 0 \,\}$$ □

Def. 5.3 The *categorization context* is a projection of a given set of schema classes onto an RDF description base. □

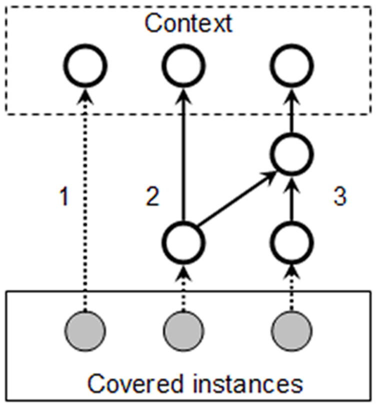


Figure 5.6 Categorization context, hierarchical distance and covered entities

We say that an instance entity *e* is *covered* by a categorization context *C*, when its hierarchical distance to the context *C* is greater than zero. A simple illustration of the coverage of instance entities is show in Figure 5.6.

Def. 5.4 Given two categorization contexts m_1 and m_2, the following *context expressions*:

- $m_1 \cap m_2$ – (intersection of contexts $\{\, e(k): e(k_1)\in m_1 \wedge e(k_2)\in m_2 \wedge k=\min(k_1,k_2) \,\}$)
- $m_1 \cup m_2$ – (union of contexts $\{\, e(k): (e(k_1)\in m_1 \vee e(k_2)\in m_2) \wedge k=\min(k_1,k_2) \,\}$)
- $m_1 \setminus m_2$ – (difference of contexts $\{\, e(k): e(k)\in m_1 \wedge \forall k_2>0: e(k_2)\notin m_2 \,\}$)

are also categorization contexts.

These are the only operations allowed on contexts. □

5.4.2. Classification into defined categories (contexts)

Classification of a document into the defined categories (contexts) requires calculating a fitness score of the document's thematic graph for each of the defined contexts. The requirements for the fitness score of the thematic graph against a given context are similar to finding the best coverage class in the bottom-up approach for the ontology- based categorization. In this case, we do not discover the best categories by building the schema structures over instance entities, but we project a context onto the thematic graph and calculate the fitness score for the covered entities. High scores should be assigned to instances that are very close to the categorization context, and that are directly covered by appropriate categories. Again, as in bottom-up class discovery, fitness score for the entities from the graph core should be higher than for the remaining entities in the graph.

The fitness score *fs* for the thematic graph *T* and context *C* is calculated using the following formula:

$$fs(C,T) = \sum_{k} w_k * h(dist_H(e_k, C)) + \sum_{n} w_{cn} * h_c(dist_H(e_{cn}, C)) \qquad (5)$$

where *k* is the number of entities in *T* covered by context *C*, n is the number of core entities in *T* covered by context *C*, e_k and e_{cn} represent all entities (normal and core) and only core entities, respectively, w_k and w_{cn} are weights of entities e_k and e_{cn}, respectively, and the functions *h* and h_c represent the importance of the entity's distance from context for normal and core entities.

In the experiments with categorization into ontology contexts, for the importance function, we used a normal distribution *N* with the mean at *1* and variance set to *2*:

$$h(dist_H(e,C)) = N_{(1,2)}(dist_H(e,C)) \tag{6}$$

We chose the normal distribution with these specific values, as it favors entities close to the context (distance up to 3) and minimizes the influence of farther entities (with distance 4 and above). The choice of diminishing the influence of classes farther than 4 relationships in the hierarchy was also influenced by the fact, that the Wikipedia category graph forms a thesaurus and not a taxonomy. Allowing too much weight for the association of an entity and a class with the hierarchical distance over 4 relationships in the thesaurus could have resulted in lower accuracy of the categorization, as classes which are not relevant could have influenced the results.

5.4.3. Composition of contexts

Ontology context provides a convenient tool for defining classification categories. However, such definition may not be sufficient to specify more complex categories that include combinations of multiple contexts.

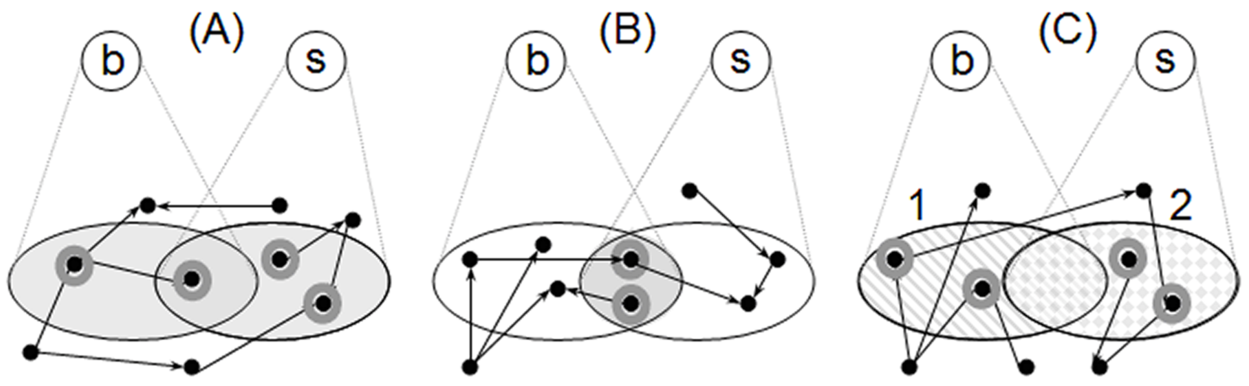


Figure 5.7 Instance graphs with selected matched entities for categories defined as intersection, union and a combination of contexts

As an example, consider the instance graph in Figure 5.7 with contexts defining “business” (b) and “sports” (s). Using the provided operators on contexts we can easily define the union, intersection, and the difference of business and sports. The union of contexts, represented by category (A), would match document graphs that include entities from the business domain, entities from the sports domain, and entities belonging to their intersection. The intersection of contexts, represented by category (B), would match document graphs that include entities that belong at the same time to both contexts. For example, this may include a sportsman who is also an entrepreneur. However, using the provided operators we are not able to define a category that would match document graphs including at the same time entities from business and sports, but not necessarily from their intersection (although it would be allowed). Such combination of contexts is represented by category (C).

To overcome this limitation imposed by defining the allowed operations on contexts, we have extended the definition of a category to include a linear combination of a number of selected categorization contexts. The extended definition of a category enables us to represent a classification category as a vector of positive context coefficients and use the vector space model for calculating the similarity (contextual fitness) of a document graph and the selected category.

For proper calculations, the context vector should be normalized so that the sum of the coefficient squares equals to 1.

Application of a vector space model for the ontology-based categorization gives the user much greater flexibility in defining a context of interest. Dynamic context definitions provide a way to provide specific, focused views of documents. With the change of user’s interest, contexts can be appropriately modified and the same documents may by instantly reclassified without the need for retraining of the classifier or providing a new set of training documents. This is one of the main strengths of the proposed training-less ontology-based categorization method.

CHAPTER 6 TEXT CATEGORIZATION EXPERIMENTAL RESULTS

In this chapter, we present the results from the experiments using our ontology-based training-less categorization method. The presented experiments were designed as comparisons of the proposed method with selected traditional text categorization techniques. We compare the accuracy achieved by each method on different document corpora. Traditional methods were trained on the pre-classified sets of documents, whereas for the ontology-based categorization, we provided a mapping of external categories to suitably selected fragments of the ontology used in our experiments. We present the results of the experiments for ontology-based categorization using both the assigned category-class mapping and the classification context approach. Finally, we provide a detailed analysis of the results and comment on the performance of the proposed method.

6.1. Experiment setup

All of our tests were performed on a machine with two 64bit Dual-Core AMD Opteron(tm) 2216 Processor operating at 2.4GHz and 8Gb of main memory. The implementation of OmniCat [53] categorization algorithms used BRAHMS [52] for storing and querying of the ontology. We utilized the previously developed algorithms for semantic associations and parts of the SPARQLeR [63] implementation to perform the categorization experiments.

We tested both proposed approaches for the ontology-based text categorization: bottom-up class discovery and top-down context projection, while the algorithm responsible for the transformation of a text document into a thematic graph remained unchanged. Sentence parsing and NLP analysis used a part-of-speech tagger [112] and chunk parser [113] by Tsurouka. The implementation of the whole system was done in C++. The categorization of a single, average document was on the order of a few seconds, and the majority of the time was devoted to the NLP analysis. We think that this amount of time is acceptable for a user to wait for the categorization of a single text document. Graph-related operations required execution time of under 1 sec and they were performed using the BRAHMS API. Other ontology storages capable of handling the Wikipedia ontology can be used for the text categorization, but the expected categorization speed of a single document may be unsatisfactory.

Our experiments included comparisons with traditional text categorization approaches. We evaluated OmniCat against two selected text categorization methods: Naïve Bayes, implemented in the BOW toolkit [71], and Support Vector Machines (SVM) from WEKA [122].

Two separate text corpora were used in all experiments. The first one contained 2,590 news documents from CNN (www.cnn.com) RSS feeds (2007-07-03 – 2007-09-04) classified into 12 categories. The second contained a subset of 2,254 documents taken from the Reuters RCV1 [69] corpora (1996-08-20 - 1996-09-02) for 6 selected categories. The details of the used text corpora are presented in Figure 6.1.

CNN Category	Training	Testing
Education	4	7
Health	91	87
money_autos	37	26
money_companies	271	275
money_taxes	15	12
Politics	171	167
science_and_space	35	27
sport_mlb	143	171
sport_nba	139	122
sport_nfl	203	222
sport_nhl	93	100
Travel	93	79
Total	**1,295**	**1,295**

Reuters Category	Training	Testing
Crime and Law Enforcement	425	448
Economic performance	73	135
Elections	129	192
Health	78	79
Religion	35	51
Sports	260	349
Total	**1,000**	**1,254**

Figure 6.1 Category details for used text corpora

The two traditional categorization methods were trained on the training set of pre-classified documents. The test sets were common for all compared text categorization algorithms.

6.1.1. Categorization ontology – Wikipedia

In the presented experiments, we used an RDF/S ontology created from the full version of English Wikipedia. The ontology was created using the modified DBpedia approach, described in Section 5.1.1, from the available XML dump dated 2008-01-03.

The extensions to the original Wikipedia source included:

- shortening of the labels that included contextual information to create literals which are more likely to appear in the text, and by this increasing the matching opportunities for entities,
- leveraging of the contextual information scheme in labels by providing relationships with confidence levels for the added context-less matching of entity phrases,

- shortening association paths that included templates by removing the blank nodes originally created by DBpedia, and
- adding specific relationships with a connecting strength level for associations involving relationships discovered directly in the text, infoboxes, templates and hypertext links.

The derived RDF/S Wikipedia ontology had the following characteristics:

- 2,096,177 unique entities in the instance base,
- 74,575,256 instance statements, out of which 44,396,459 were hypertext references,
- 4,805,215 unique literals used as matching phrases for entities,
- 6,351,497 literal statements, giving on average 2.29 literals assigned to an entity,
- 348,171 Wikipedia categories encoded as schema classes,
- 601,796 statements on the schema level describing Wikipedia category graph,
- 5,943,872 type classification statements, giving on average single entity classified into 2.84 categories.

The Wikipedia ontology used in our experiments required a BRAHMS snapshot of size 7.6GB on a 64-bit system.

6.1.2. Category mapping to ontology

Categories in both corpora can be aligned with a few general Wikipedia categories. This provides an approximation for the proper mapping of categories to the internal Wikipedia category structure. For the top-down classification approach, such alignment in most cases was sufficient, as the projection of a context onto an instance base does not require intermediate classes to be included in the context definition.

The bottom-up approach for category discovery does not require any category mapping to perform classification into the taxonomy included in the ontology. However, such mapping is necessary for assigning categories that are defined externally to the ontology. Moreover, in the implemented approach, we search for the intersection of the top assigned classes with the category defined as a set of classes. This requires incorporating Wikipedia's subcategories into class definitions. During the creation of mappings that included subcategories, we had to take into account that the Wikipedia category graph forms a thesaurus and not a taxonomy. The subcategory relation is not equivalent to the subclass relation. We had to limit the search depth for the included subcategories to minimize the number of categories which are unrelated to the top class. Figure 6.2 presents the alignment of CNN categories and Wikipedia top classes together with the number of subclasses included into the category definitions.

CNN category	Wikipedia root classes	Number of subcategories
education	Category:Education	1467
health	Category:Health	1505
money_autos	Category:Automobiles	1350
money_companies	Category:Business Category:Economics Category:Stock_market	2509
money_taxes	Category:Accountancy Category:Taxation	146
politics	Category:Politics Category:Politicians	7746
science_and_space	Category:Science Category:Space	833
sport_mlb	Category:Baseball Category:Major_League_Baseball	1710
sport_nba	Category:Basketball Category:National_Basketball_Association	2438
sport_nfl	Category:Football Category:National_Football_League	8251
sport_nhl	Category:Hockey Category:National_Hockey_League	1858
travel	Category:Travel	714

Figure 6.2 Alignment of CNN categories and top Wikipedia categories

Additionally, for one of the experiments we prepared a test and train set of documents directly from the English Wikipedia. To test the quality of the semi-automatically generated mapping we prepared a testing and training set of document that included page content of entries categorized to classes included in the category definitions. All of the selected categories for the CNN mapping cover over 400,000 entries in Wikipedia. To limit the size of such prepared corpora to match the volume of other corpora used for experiments, we decided to randomly select up to 2,000 pages for each CNN category from the documents classified in Wikipedia under the mapped categories. We prepared 10 different sets of Wikipedia documents to test the consistency of our categorization method. They were used both as the training and testing sets for traditional text categorization methods. We also examined the accuracy of their categorization using OmniCat with the Wikipedia ontology.

The categories from the selected subset of the Reuters corpora can also be aligned with the high level categories in Wikipedia. The selected alignment is presented in Figure 6.3.

Reuters category	**Wikipedia category**
Crime and Law Enforcement	Category:Law Category:Crime
Economic Performance	Category:Business Category:Economics
Elections	Category:Elections Category:Politics
Health	Category:Health Category:Medicine
Religion	Category:Religion Category:Theology
Sports	Category:Sports

Figure 6.3 Alignment of Reuters categories and top Wikipedia categories

For the creation of the Reuters category mapping, we arbitrarily limited the depth of the included subcategories to 3. It was empirically selected following numerous categorization trials of the ontology-based categorization method with different depth values.

6.2. Results of experiments with bottom-up class discovery

We performed the following experiments using the bottom-up class discovery and matching it with the provided category mappings:

- comparison of the categorization accuracy on CNN corpora using OmniCat and BOW's Naïve Bayes trained (a) on the CNN training set and (b) on the sets of selected pages from Wikipedia,
- quality estimation of the created CNN category mapping by categorizing the prepared Wikipedia corpora using BOW's Naïve Bayes trained on Wikipedia corpora,
- comparison of the categorization accuracy on selected Reuters corpora using OmniCat, BOW's Naïve Bayes and SVM trained on the Reuters training set

6.2.1. Comparison of OmniCat and BOW on CNN categories

In our first experiment we compared the OmniCat bottom-up algorithm with classical Naïve Bayes approach. Detail categorization results of OmniCat, Naïve Bayes (BOW) trained on the Wikipedia corpora and Naïve Bayes (BOW) trained on the CNN corpora and presented respectively in Figure 6.4, Figure 6.5, and Figure 6.6.

Category		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**total**	**correct**
0	**education**	**6**	.	.	.	.	1	.	.	.	.	.	.	.	7	85.71%
1	**health**	2	**70**	.	3	.	3	3	.	.	.	.	3	.	87	80.46%
2	**money_autos**	.	.	**17**	8	.	.	1	.	.	.	.	.	.	26	65.38%
3	**money_companies**	.	20	10	**213**	19	2	5	1	.	.	.	5	.	275	77.45%
4	**money_taxes**	.	.	.	3	**8**	1	.	.	.	.	.	.	.	12	66.67%
5	**politics**	.	6	.	8	.	**148**	3	.	.	.	.	2	.	167	88.62%
6	**science_and_space**	1	1	.	1	.	1	**21**	.	.	.	.	2	.	27	77.78%
7	**sport_mlb**	.	2	.	6	.	3	1	**153**	.	.	.	6	.	171	89.47%
8	**sport_nba**	.	3	.	7	.	12	3	.	**91**	1	.	4	1	122	74.59%
9	**sport_nfl**	.	3	1	7	.	16	4	.	.	**185**	.	6	.	222	83.33%
10	**sport_nhl**	.	.	1	7	.	3	2	.	1	3	**77**	6	.	100	77.00%
11	**travel**	.	5	.	7	.	9	10	.	.	.	.	**48**	.	79	60.67%
12	**unknown**	.	.	.	.	.	.	.	.	.	.	.	.	.	0	0.00%
	Classified documents:	1295														
	Correctly classified:	1037														
	Achieved accuracy:	80.08%														

Figure 6.4 OmniCat bottom-up categorization of CNN documents to prepared category mapping

Correct: 949 out of 1295 (73.28 percent accuracy); Confusion details, row is actual, column is predicted

Category		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**total**	**Correct**
0	**education**	**7**	.	.	.	.	.	.	.	.	.	.	.	7	100.00%
1	**health**	23	**55**	.	1	1	.	3	.	.	.	.	4	87	63.22%
2	**money_autos**	.	.	**23**	3	.	.	.	.	.	.	.	.	26	88.46%
3	**money_companies**	1	11	16	**169**	74	.	.	.	.	.	.	4	275	61.45%
4	**money_taxes**	.	.	.	.	**12**	.	.	.	.	.	.	.	12	100.00%
5	**politics**	11	4	.	2	40	**100**	.	.	.	.	.	10	167	59.88%
6	**science_and_space**	1	.	.	.	.	.	**22**	.	.	.	.	4	27	81.48%
7	**sport_mlb**	1	1	.	3	5	.	.	**155**	.	.	.	6	171	90.64%
8	**sport_nba**	2	.	.	3	4	.	.	.	**99**	1	.	13	122	81.15%
9	**sport_nfl**	13	1	.	9	7	2	.	.	8	**160**	.	22	222	72.07%
10	**sport_nhl**	5	.	1	7	5	.	.	.	2	.	**75**	5	100	75.00%
11	**travel**	1	1	.	2	3	.	.	.	.	.	.	**72**	79	91.14%
	Percent accuracy average: 73.28 stderr 0.00														

Figure 6.5 BOW categorization of CNN documents using the Wikipeda corpora as training

Correct: 1220 out of 1295 (94.21 percent accuracy); Confusion details, row is actual, column is predicted

Category		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**total**	**Correct**
0	**education**	**2**	.	.	.	.	2	.	.	.	.	.	.	4	50.00%
1	**health**	.	**80**	.	4	.	2	.	.	.	2	.	3	91	87.91%
2	**money_autos**	.	.	**16**	20	.	.	.	.	.	.	.	1	37	43.24%
3	**money_companies**	.	1	4	**263**	.	2	.	.	.	.	.	1	271	97.05%
4	**money_taxes**	.	.	.	9	**4**	2	.	.	.	.	.	.	15	26.67%
5	**politics**	.	2	.	.	.	**169**	.	.	.	.	.	.	172	98.83%
6	**science_and_space**	.	.	.	.	.	.	**33**	.	1	.	.	1	35	94.29%
7	**sport_mlb**	.	.	.	1	.	.	.	**140**	.	2	.	.	143	97.90%
8	**sport_nba**	.	.	.	.	.	.	.	1	**135**	3	.	.	139	97.12%
9	**sport_nfl**	.	.	.	.	.	1	.	.	.	**202**	.	.	203	99.51%
10	**sport_nhl**	.	.	.	1	.	.	.	.	2	.	**90**	.	93	96.77%
11	**travel**	.	1	.	2	.	4	.	.	.	.	.	**86**	93	92.47%
	Percent accuracy average: 94.21 stderr 0.00														

Figure 6.6 BOW categorization of CNN document using CNN training set

The OmniCat ontology-based categorization method that used the bottom-up approach for finding the best ontological categorization reached the accuracy of 80% on CNN corpora with the supplied CNN-Wikipedia category mapping. Our method did not require any training. The discovery of the best fitting ontological categories did not require any additional knowledge beside the ontology used for categorization. Assigning of the externally defined categories required providing their mapping to the ontology taxonomy.

The Naïve Bayes implementation from BOW required providing a training set of pre-classified documents. On the categorization of CNN documents, while using the Wikipedia training set, the BOW's Naïve Bayes classifier achieved only 73% accuracy. Using the training and testing documents from the same source, the CNN corpora, the BOW classifier achieved accuracy slightly over 94%. For this experiment, no intermediate mapping was required.

The significant difference of the results in the two experiments using Naïve Bayes suggests that the content of Wikipedia documents, although selected according to the created category mapping, significantly differs from the news articles collected from CNN. Such an observation may be valid, as documents in CNN are news stories, and articles in Wikipedia describe encyclopedic knowledge and are written in a different style using a different vocabulary.

6.2.2. Quality estimation of CNN-Wikipedia category mapping

The second experiment was performed to indirectly estimate the quality of the prepared CNN category mapping to Wikipedia classes. We performed the categorization of the selected Wikipedia articles using Naïve Bayes (BOW) trained on Wikipedia corpora. The categorization results are presented in Figure 6.7.

Correct: 18451 out of 22154 (83.29 percent accuracy); Confusion details, row is actual, column is predicted

	Category	0	1	2	3	4	5	6	7	8	9	10	11	total	Correct
0	**education**	**1655**	12	1	24	42	60	109	.	1	1	.	47	1952	84.78%
1	**health**	232	**1243**	6	42	26	30	198	.	1	4	3	125	1910	65.08%
2	**money_autos**	11	3	**1665**	63	5	13	37	.	.	5	1	132	1935	86.05%
3	**money_companies**	73	12	36	**1303**	147	33	42	.	1	3	3	220	1873	69.57%
4	**money_taxes**	4	.	.	4	**869**	31	4	1	.	3	1	11	928	93.64%
5	**politics**	174	3	2	39	104	**1425**	12	1	1	14	4	116	1868	76.28%
6	**science_and_spac**	289	131	24	38	58	24	**1218**	.	.	.	1	147	1930	63.11%
7	**sport_mlb**	2	.	.	3	2	.	.	**1933**	27	29	8	9	2013	96.03%
8	**sport_nba**	5	.	.	26	.	6	.	13	**1883**	37	14	14	1998	94.24%
9	**sport_nfl**	4	.	1	11	.	9	.	5	18	**1885**	12	9	1954	96.47%
10	**sport_nhl**	8	.	1	18	.	1	.	2	49	39	**1742**	43	1903	91.54%
11	**travel**	48	24	21	71	34	17	39	1	2	2	1	**1630**	1890	86.24%
	Percent accuracy average: 83.29 stderr 0.00														

Figure 6.7 BOW categorization of Wikipedia documents using Wikipedia mapping training set

In this experiment, BOW's Naïve Bayes achieved accuracy slightly over 83%. This result suggests the confidence level of quality of the created mapping. On the detailed categorization results, we can notice that some categories were defined too broadly, and during the categorization process they were assigned to multiple documents from other categories. These included "education", "travel", and "politics". On the other hand, the results on the "science and space" category suggest that it was not defined strictly enough or that the documents from this category often combined multiple domains, where "science and space" was not the dominant one.

6.2.3. Comparison of OmniCat, Naïve Bayes and SVM on Reuters categories

The last experiment in this series was performed using the selected subset of the Reuters corpora. We compared the classification accuracy achieved by OmniCat, Naïve Bayes (BOW) and SVM. The Naïve Bayes and SVM classifiers were trained on the Reuters training set. OmniCat used a provided category mapping between Reuters categories and Wikipedia classes. The results of the categorization accuracy comparison are presented in Figure 6.8.

	BOW	**SVM**	**ONTO**
Crime and Law Enforcement	93.1%	95.8%	87.1%
Economic Performance	88.9%	85.9%	85.9%
Elections	96.9%	91.7%	90.1%
Health	70.9%	53.2%	59.5%
Religion	33.3%	23.5%	58.8%
Sports	97.1%	96.6%	92.6%
Total	**90.5%**	**88.7%**	**86.1%**

Figure 6.8 Categorization results of BOW, SVM and OmniCat on Reuters corpora

The two selected traditional categorization methods performed better in this test. They achieved 90.5% (BOW's Naïve Bayes) and 88.7% (SVM). OntoCat achieved only 86.1%, which is still within the range of good text categorization methods. It is worth noting, that OmniCat's accuracy on the "religion" category was significantly better than the other two methods. We believe that the main reason for it was a well-developed category schema in Wikipedia for religion and theology.

6.3. Results of experiments with top-down context projection

We performed the following experiments using the top-down context projection of the aligned Wikipedia categories:

- categorization accuracy of the OmniCat on the CNN corpora,
- categorization accuracy of the OmniCat on the subset of Reuters corpora,
- comparison of the categorization accuracy on CNN corpora of BOW's Naïve Bayes, SVM, and both algorithms in OmniCat, when Naïve Bayes and SVM were trained on the CNN training set,
- comparison of the categorization accuracy on the selected subset of Reuters corpora of BOW's Naïve Bayes, SVM, and both algorithms in OmniCat, when Naïve Bayes and SVM were trained on the Reuters training set,

6.3.1. OmniCat context on CNN

The first experiment with the use of categorization contexts was performed on the CNN corpora. We utilized opportunities offered by the context algebra to better accommodate the categories for the specificity of news documents. The categorization of a news article tends to concentrate on people who are mentioned in it. It means that for news categorization, entities representing people should be of higher importance. For this experiment, we defined categories as the union of the original context with the intersection of the people category and the original context:

$$Context_{mod} = Context \cup (Context \cap Person)$$

Using such approach, we significantly reduced the hierarchical distance of entities representing people in the specific context to the context itself. Previously, the distance of such entities was determined by the placement of original classes in the hierarchy, while in this case it was reduced to 1, as *Person* was matched as the closest class in context. The detailed results from this classification experiment are presented in Figure 6.9.

Category		1	2	3	4	5	6	7	8	9	10	11	12	correct
1	**education**	**6**	.	.	.	.	1	.	.	.	.	.	.	85.71%
2	**health**	2	**53**	.	4	1	3	24	.	.	.	.	.	60.92%
3	**money_autos**	.	.	**21**	2	.	1	2	.	.	.	.	.	80.77%
4	**money_companies**	.	19	16	**208**	1	8	22	.	1	.	.	.	75.64%
5	**money_taxes**	.	.	.	5	**4**	2	1	.	.	.	.	.	33.33%
6	**politics**	1	.	.	11	.	**152**	3	.	.	.	.	.	91.02%
7	**science_and_space**	.	.	.	2	.	1	**24**	.	.	.	.	.	88.89%
8	**sport_mlb**	.	1	.	6	.	5	.	**159**	.	.	.	.	92.98%
9	**sport_nba**	.	2	3	5	.	4	3	.	**105**	.	.	.	86.07%
10	**sport_nfl**	.	.	2	8	.	6	8	.	.	**198**	.	.	89.19%
11	**sport_nhl**	.	2	.	8	.	5	3	.	1	.	**80**	1	80.00%
12	**unknown**	.	.	.	.	.	.	.	.	.	.	.	.	
	Classified documents:	1216												
	Correctly classified:	1010												
	Achieved accuracy:	83.1%												

Figure 6.9 Categorization results of OmniCat with contexts on CNN corpora

The OmniCat categorizer with the contextual definition of categories achieved the 83% accuracy. We believe that this moderate improvement can be attributed to the modified definition of categories.

6.3.2. OmniCat context on Reuters

In this test, we perform the categorization of a selected Reuters corpora using OmniCat with the categories defined as contexts. For this experiment, we did not modify the contexts definitions and used the selected root classes from Wikipedia. The detailed results are presented in Figure 6.10.

Category		1	2	3	4	5	6	7	Correct
1	**Crime and Law Enforcement**	**333**	39	29	25	9	8	5	74.3%
2	**Economic Performance**	7	**112**	5	5	6	.	.	83.0%
3	**Elections**	5	3	**178**	6	.	.	.	92.7%
4	**Health**	.	7	12	**52**	.	7	1	65.8%
5	**Religion**	8	3	6	3	**29**	.	2	56.9%
6	**Sports**	6	11	.	13	2	**314**	3	90.0%
7	**unknown**	.	.	.	.	.	.	.	
	Classified documents: 1254								
	Correctly classified: 1018								
	Achieved accuracy: 81.2%								

Figure 6.10 Categorization results of OmniCat with contexts on Reuters corpora

Our OmniCat categorization method achieved the average accuracy of 81%. We consider this result to be in the acceptable range for the text categorization methods. Still, we should stress that OmniCat achieved such result without providing a training set of pre-classified documents.

6.3.3. Comparison of OmniCat, Naïve Bayes and SVM on CNN and Reuters corpora

In this section, we present the comparison of the results achieved by both of our categorization schemes against the BOW's Naïve Bayes and SVM categorization methods. The summary of the results from multiple experiments are presented in Figure 6.11.

CNN	BOW	SVM	Onto	OmniCat
education	28.5%	28.6%	71.4%	85.7%
health	96.5%	95.4%	54.0%	60.9%
money_autos	46.1%	57.7%	73.1%	80.8%
money_companies	97.5%	96.4%	71.6%	75.6%
money_taxes	41.7%	50.0%	16.7%	33.3%
politics	98.2%	98.8%	88.6%	91.0%
science_and_space	100.0%	100.0%	88.9%	88.9%
sport_mlb	98.8%	95.3%	92.4%	93.0%
sport_nba	95.9%	95.9%	83.6%	86.1%
sport_nfl	100.0%	99.1%	87.8%	89.2%
sport_nhl	97.0%	97.0%	78.0%	80.0%
Total	**95.8%**	**95.4%**	**80.2%**	**83.1%**

Reuters	BOW	SVM	Onto	OmniCat
Crime and Law Enforcement	93.1%	95.8%	87.1%	74.3%
Economic performance	88.9%	85.9%	85.9%	83.0%
Elections	96.9%	91.7%	90.1%	92.7%
Health	70.9%	53.2%	59.5%	65.8%
Religion	33.3%	23.5%	58.8%	56.9%
Sports	97.1%	96.6%	92.6%	90.0%
Total	**90.5%**	**88.7%**	**86.1%**	**81.2%**

Figure 6.11 Comparison of categorization results of Naïve Bayes (BOW), SVM and both OmniCat algorithms on CNN and Reuters corpora

The side by side comparison of the categorization accuracy results achieved by Naïve Bayes, SVM and both OmniCat categorization schemes shows that our method can reach good accuracy levels, comparable with the traditional categorization methods. So far, it cannot achieve the accuracy provided by traditional text categorization methods, but as a newly proposed categorization approach, it may be refined and gradually improved in the future.

Note that the achieved good results by our ontology-based categorization method did not require any training set. To perform the classification, we needed only to provide category definitions as ontology class mappings or categorization contexts. Additionally, with the possibility of dynamic changes to the categorization contexts, our method allows to instantly reclassify the same document corpora according to the newly expressed user interest. This approach allows for flexibility in looking at the documents from different perspectives and defining the variety of the categorization interests that are not limited by the existence of a proper training set.

6.4. Analysis of the results

Overall, both of the proposed approaches for categorization in our ontology-based method achieved very good results in the performed tests. Compared to the two selected traditional categorization methods, the difference in the achieved accuracy was within 9% for Reuters and 12% for the CNN corpora.

Both traditional categorization methods used training sets of pre-classified documents that came from the same source as the classified documents. This means that the documents for training and classification had similar format, wording, and other features that may influence the similarity. On the other hand, OmniCat categorized documents according to a given ontology, which provided somewhat different descriptions of domains and the categories than the used corpora, and the definition of the external categories had to be approximated by the selected ontology fragments.

The achieved results are especially promising, as our categorization method did not use training sets of documents. The classifier did not require any training, and the modification of the categories or the introduction of new categories would not require rebuilding or recreating of the classifier. It would be ready to use as soon as the new category definitions were provided.

Misclassified documents are one of the best sources of knowledge about flaws and imperfections of the used categorization method. We analyzed the incorrectly classified documents together with the created thematic graphs and identified the following common sources of misclassifications:

- the created mapping between the external categories and Wikipedia classes were too broad and imprecise,
- the prepared categorization contexts used the Wikipedia category hierarchy, which not always reflects the topics covered in the news,
- the imbalanced structure of the Wikipedia articles and the difference in coverage of the domains biases categorization towards densely connected ontology fragments,
- disagreement between the category describing the majority of the document content, and the originally assigned category based on user's *perceived interest*.

The mapping between external categories defined in the corpora and the category hierarchy in Wikipedia is inherently imprecise. It serves as an approximation of a user defined category to taxonomy classes provided in the ontology. The mappings were rooted to high level categories in Wikipedia that were manually aligned to the given corpora's categories. For extending a mapping by appropriate subcategories, we used the associations present in the category graph in Wikipedia. Instead of an expected taxonomy, it forms a thesaurus. Using the thesaurus subcategory associations, we added several classes to the mapping definition of a category. This included the subcategories that were both strongly and only partially related to the original category. Using this approach of extending the category mapping, in some cases caused blurring of the original understanding of a category due to the addition of unrelated classes.

Wikipedia is a free electronic encyclopedia that everybody can edit. In describes numerous domains, and the coverage and precision of the provided entries depends on the number of devoted users who add and edit the content, and the quality of knowledge introduced by them. Groups of highly interested and motivated users may create articles and content in a given domain, making it a highly connected fragment of an ontology. One of such examples is the biomedical domain in Wikipedia. When a document contains entities from such a domain, it is more probable that they may dominate the categorization process. This feature is strongly related to the ontology itself and the ontology quality, and there is no simple compensation for it.

The last misclassification source is connected to the user's *perceived interest* of a category. It is a disagreement between the majority of the document's content represented by the recognized entities and the originally assigned category. The ontology-based categorization relies on the recognized entities and the knowledge provided by the ontology to classify the document. The created Wikipedia-based ontology contains encyclopedic knowledge which describes the basic facts and their relationships. It does not favor any specific types of entities such as people, companies, or places. On the other hand, people may decide the category of a document based on a single, yet perceived as highly important entity.

As an example, consider an article about cardio-vascular health problems of a certain politician. From a reader's perspective, the article belongs to politics, as the politician is the main point of interest. However, the analysis of the document shows that the majority of its content is about the disease, treatment, or perhaps the recovery. The ontology-based categorization method most probably would focus the analysis on the entities form this domain, and the politician would not become one of the core entities. This would result in the final classification of the document into the health category, and not politics, as it was originally assigned by user.

The introduced categorization contexts and their combinations provide a partial solution for this problem. In the experiments described in Section 6.3.1, the use of specially defined contexts helped achieve a better accuracy. Contexts present views of a document through specific interest lenses. Some entities and relationships may become more important, while other can disappear. The definitions of categorization contexts still can be extended to capture more of the user's interest. The current definitions help in the categorization, but do not completely solve the problem of capturing the user's perceived interest.

Despite the imprecise coverage of the news topics by the Wikipedia-based created category mappings and the categorization contexts, the imbalance in the coverage of some domains, and the difference of the encyclopedic knowledge and perceived interest in categories, our ontology-based training-less categorization method was able to achieve comparable results to the selected traditional categorization methods. An important aspect of the proposed method is that it does not rely on the training set of pre-classified documents. Furthermore, with the change of interest, new context definitions can be provided, and without changing the ontology or rebuilding the classifier, we can instantly begin categorization of documents to the new categories.

CHAPTER 7 CONCLUSIONS AND FUTURE WORK

The major part of the research presented in this dissertation focused on the problem of text categorization with the use of ontology. We presented a prototype of the ontology-based training-less text categorization, called OmniCat. Together with OmniCat, we presented two additional contributions that were utilized in creating the ontology-based text categorizer: a high-performance BRAHMS RDF/S storage and the SPARQLeR query language.

The proposed ontology-based text categorization method that does not rely on the training set is a novel approach to the automatic text categorization. Although the use of ontology and other sources of external knowledge was previously proposed by many approaches, such as presented in [23] [32] [75] [89] [127], all of them depended on the presence of a pre-classified set of training documents. We decided to directly leverage the knowledge present in the ontology in order to identify relevant topics for the analyzed documents, based on the recognized named entities and discovered semantic associations, and later perform the document categorization utilizing this information and the ontology taxonomy.

Relinquishing the need for a training set required the creation of an alternative method for defining classification categories. As the proposed text categorization method is based on the ontology, the natural consequence was to define categories as ontology fragments. To allow a more focused categorization, we introduced the notion of a categorization context and a context projection. Using these definitions, complex categories can be expressed as combinations of contexts, making it a very flexible and expressive approach.

The results achieved in our text categorization with the use of good quality ontology are encouraging. In our tests we used an ontology derived from a full version of English language Wikipedia. The average accuracy achieved in the categorization of the CNN and the Reuters corpora was at or above 80%. This proves the applicability of an ontology for the text categorization purposes. However, there is room for improvement, as traditional text categorization methods can achieve better results.

Another advantage of OmniCat is that is allows dynamic definition of classification categories. With a suitable change of categorization contexts, we can immediately shift the categorization focus to other classes and features without having to modify the classifier. There is no training set included, so no retraining of the classifier is required. Additionally, using different descriptions of categorization contexts, we are able to classify the same documents in multiple ways, according to the changing interests of the users.

The proposed categorization method is not without weaknesses. The process of automatic text categorization can be viewed as a search for important and distinguishing features that help to identify the category of a document. The proposed text categorization method relies on the factual knowledge from the ontology that does not always match the user's perceived interests. Focusing on the document content supported by the background knowledge, may miss other important contextual interpretations. Categorization contexts were introduced to overcome this issue. They allow focusing the categorization on the areas in the ontology that are important for the category, but may be underrepresented in the document. However, in some cases, the knowledge induced by the ontology can make one domain to dominate the created thematic graph, diminishing the importance of other included topics. It happens when one of the topics recognized in the document belongs to a well-developed and highly-connected fragment of the ontology. Entities from such category may create a heavily connected component that dominates the core of the thematic graph and pushes out entities related to other classes to the outskirts of the graph, or even prevents them to be included in the thematic graph. Such imbalances in the development of different ontology areas not always can be compensated with the use of categorization contexts.

Despite some limitations of the proposed ontology-based text categorization method, we believe that as rich and comprehensive domain ontologies become more available, the proposed approach may be a suitable alternative to the traditional text categorization methods.

Our successful use of the English language Wikipedia in the performed experiments demonstrated a significant value of the represented knowledge for the text categorization task. Multiple language versions of Wikipedia open new possibilities for cross-language ontology-based categorization. An initial approach for using multiple Wikipedias from different languages for text categorization is described in [83]. We believe that our ontology-based text categorization method can be successfully applied for cross lingual text categorization. The OmniCat method relies on entities, their connections, and the assigned categories. Labels in Wikipedia that describe entities in a specific language are used only to identify entities in the document. Once the entities in a given document are recognized, we can utilize the encoded mapping of the entities and their categories across different languages in Wikipedia, and transform the previously created thematic graph to a different language in a straightforward way. This enables the presentation of the main domain of the document, as well as its final categorization in a language different than the original language of the document. In the near future we plan to investigate such possibility.

The categorization in the OmniCat method is based on establishing the semantic similarity between a thematic graph that represents the analyzed document and a category defined as an appropriate ontology fragment. Transformation of an unstructured text document into a thematic graph is a necessary intermediate step, so we could perform a comparison of two graphs and calculation of similarity. One of the directions in the future work is to explore the potential offered by this method for semantic search for ontologies or categorization of ontologies with the use of a reference ontology.

REFERENCES

[1] British National Corpus, version 3 (BNC XML Edition). 2007. Distributed by Oxford University Computing Services on behalf of the BNC Consortium. URL: http://www.natcorp.ox.ac.uk/.

[2] BYU Corpus of American English. 360 million words (1990-present). Available online at http://www.americancorpus.org. Created by Mark Davies (Brigham Young University), 2008.:

[3] Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium. Nature Genet. **25** (2000) 25-29

[4] Glycomics - Bioinformatics for Glycan Expression. Integrated Technology Resource for Biomedical Glycomics: Technological Research and Development Project IV, A Project funded by NIH, http://lsdis.cs.uga.edu/projects/glycomics/

[5] PubMed: The National Library of Medicine. Bethesda, MD

[6] SemDis API ver. 0.3. http://lsdis.cs.uga.edu/projects/semdis/api/

[7] Wikipedia: The Free Encyclopedia. Wikimedia Foundation.: http://en.wikipedia.org (2004)

[8] WordNet: An Electronic Lexical Database. The MIT Press (1998)

[9] Abiteboul, S., Goldman, R., McHugh, J., Vassalos, V., Zhuge, Y.: Views for Semistructured Data. Workshop on Management of Semistructured Data, Tucson, AZ, USA (1997)

[10] Al-Khateeb, F., Baget, J.-F., Euzenat, J.: Complex Path Queries for RDF Graphs. Poster Session at the 4th International Semantic Web Conference (ISWC2005), Galway, Ireland (2005)

[11] Aleman-Meza, B., Burns, P., Eavenson, M., Palaniswami, D., Sheth, A.P.: An Ontological Approach to the Document Access Problem of Insider Threat. IEEE International Conference on Intelligence and Security Informatics (ISI-2005), Atlanta, Georgia, USA (2005)

[12] Aleman-Meza, B., Halaschek, C., Sheth, A., Arpinar, I.B., Sannapareddy, G.: SWETO: Large-Scale Semantic Web Test-bed. 16th International Conference on Software

Engineering and Knowledge Engineering (SEKE2004): Workshop on Ontology in Action, Banff, Canada (2004) 490-493

[13] Aleman-Meza, B., et al.: Semantic Analytics on Social Networks: Experiences in Addressing the Problem of Conflict of Interest Detection. 15th International World Wide Web Conference, WWW'06. ACM Press, New York, NY, Edinburgh, Scotland (May 23-26, 2006) 407-416

[14] Angles, R., Gutierrez, C.: Querying RDF Data from a Graph Database Perspective. 2nd. European Semantic Web Conference (ESWC2005), Heraklion, Greece (2005)

[15] Angles, R., Gutierrez, C., Hayes, J.: RDF Query Languages Need Support for Graph Properties. Universidad de Chile, Technical Report TR/DCC-2004-3 (2004)

[16] Anyanwu, K., Sheth, A.: r-Queries: Enabling Querying for Semantic Associations on the Semantic Web. The Twelfth International World Wide Web Conference, Budapest, Hungary (2003)

[17] Auer, S., Lehmann, J.: What have Innsbruck and Leipzig in common? Extracting Semantics from Wiki Content.: European Semantic Web Conference (ESWC'07). Springer, Innsbruck, Austria (2007) 503-517

[18] Beckett, D.: Creating additional storage hashes. (2003) redland-dev - Redland development mailing list

[19] Beckett, D.: The Design and Implementation of the Redland RDF Application Framework. Tenth International World Wide Web Conference, Vol. 1-58113-348-0/01/0005. ACM, Hong Kong (2001)

[20] Beckett, D.: Rasqal RDF Query Library. http://librdf.org/rasqal/

[21] Berners-Lee, T., Cailliau, R.: WorldWideWeb: Proposal for a HyperText Project. http://www.w3.org/Proposal.html (1990)

[22] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American (2001)

[23] Bloehdorn, S., Hotho, A.: Text Classification by Boosting Weak Learners based on Terms and Concepts. 4th IEEE International Conference on Data Mining (ICDM'04) (2004)

[24] Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. In: McBride, B. (ed.): http://www.w3.org/TR/rdf-schema/ (10 Feb 2004)

[25] Broekstra, J., Kampman, A.: SeRQL: A Second Generation RDF Query Language. SWAD-Europe Workshop on Semantic Web Storage and Retrieval, Amsterdam, Netherlands (2003)

[26] Broekstra, J., Kampman, A., Harmelen, F.v.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. International Semantic Web Conference, Sardinia, Italy (2002)

[27] Cardoso-Cachopo, A., Oliveira, A.L.: An Empirical Comparison of Text Categorization Methods. String Processing and Information Retrieval. SpringerLink (2003)

[28] Consens, M., Mendelzon, A.O.: Graphlog: a visual formalism for real life recursion. ACM Symposium On Principles of Database Systems (1990) 404-416

[29] Cruz, I.F., Mendelzon, A.O., Wood, P.T.: G+: Recursive queries without recursion. 2nd International Conference on Expert Database Systems (1988) 355-368

[30] Decker, S., Sintek, M., Nejdl, W.: The Model-Theoretic Semantics of TRIPLE. Technical Report (2002)

[31] Deerwester, S., Dumais, S., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. Journal of the Society for Information Science (1990) **41** (1990) 391-407

[32] Gabrilovich, E., Markovitch, S.: Overcoming the Brittleness Bottleneck using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. 21th National Conference on Artificial Intelligence, Boston, MA, USA (2006)

[33] Greenbaum, S.: ICE: the International Corpus of English. English Today **28** (1991) 3-7

[34] Gruber, T.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition **5** (1993) 199-220, 1993

[35] Guarino, N.: Towards a Formal Evaluation of Ontology Quality. IEEE Intelligent Systems (2004) 1541-1672, 2004

[36] Guarino, N., Welty, C.: An Overview of OntoClean. In: Staab, S., Studer, R. (eds.): Handbook on Ontologies. Springer Verlag (2004) 151-159

[37] Guha, R., McCool, R., Fikes, R.: Contexts for the Semantic Web. 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan (2004)

[38] Guo, Y., Pan, Z., Heflin, J.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. Third International Semantic Web Conference, Vol. LNCS 3298. Spinger, Hiroshima, Japan (2004) 274-288

[39] Haarslev, V., Möller, R.: Racer: An OWL Reasoning Agent for the Semantic Web. International Workshop on Applications, Products and Services of Web-based Support Systems, at the 2003 IEEE/WIC International Conference on Web Intelligence, Halifax, Canada (2003)

[40] Hammond, B., Sheth, A.P., Kochut, K.J.: Semantic Enhancement Engine: A Modular Document Enhancement Platform for Semantic Applications over Heterogeneous Content. Real World Semantic Web Applications, IOS Press, 2002 (2002)

[41] Han, E.-H., Karypis, G.: Centroid-Based Document Classification: Analysis Experimental Results. Principles of Data Mining and Knowledge Discovery (2000) 424-431

[42] Handschuh, S., Staab, S.: CREAM CREAting Metadata for the Semantic Web. Computer Networks **42** (2003) 579-598

[43] Handschuh, S., Staab, S., Studer, R.: Leveraging Metadata Creation for the Semantic Web with CREAM. 26th Annual German Conference on Artificial Intelligence, Hamburg, Germany (2003) 19-33

[44] Hartigan, J.A., Wong, M.A.: A K-Means Clustering Algorithm. Applied Statistics **28** (1979) 100-108

[45] Hartley, H.O., Rao, J.N.K.: Classification and estimation in analysis of variance problems. Review of International Statistical Institute **36** (1968) 141-147

[46] Hassell, J., Aleman-Meza, B., Arpinar, I.B.: Ontology-Driven Automatic Entity Disambiguation in Unstructured Text. 5th International Semantic Web Conference (ISWC-2006), Athens, GA, USA (November 6-9, 2006)

[47] Hayes, P.: RDF Semantics. In: McBride, B. (ed.): http://www.w3.org/TR/rdf-mt/ (10 Feb 2004)

[48] He, Q., Qiu, L., Zhao, G., Wang, S.: Text Categorization Based on Domain Ontology. Fifth International Conference on Web Information Systems – WISE 2004, Brisbane, Australia (2004)

[49] Helenius, A., Aebi, M.: Roles of N-Linked Glycans in the Endoplasmic Reticulum. Annual Review of Biochemistry **2004, 73** (2004) 1019-1049

[50] Horrocks, I., Patel-Schneider, P.F.: Reducing OWL entailment to description logic satisfiability. Journal of Web Semantics **1** (2004) 345-357

[51] Hotho, A., Staab, S., Stumme, G.: Ontologies improve text document clustering. 3rd Third IEEE International Conference on Data Mining (ICDM 2003) (2003)

[52] Janik, M., Kochut, K.J.: BRAHMS: A WorkBench RDF Store And High Performance Memory System for Semantic Association Discovery. Fourth International Semantic Web Conference (ISWC 2005), Galway, Ireland (2005)

[53] Janik, M., Kochut, K.J.: OmniCat: Automatic Text Classification with Dynamically Defined Categories. UGA Computer Science Technical Report No. 08.001, Athens, GA, USA (2008)

[54] Janik, M., Kochut, K.J.: Training-less Ontology-based Text Categorization. Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR 2008) at the 30th European Conference on Information Retrieval (ECIR'08), Glasgow, Scotland (2008)

[55] Janik, M., Kochut, K.J.: Wikipedia in Action: Ontological Knowledge in Text Categorization. 2nd International Conference on Semantic Computing (ICSC). [to appear], Santa Clara, CA, USA (2008)

[56] Jardine, N., Sibson, R.: The construction of hierarchic and non-hierarchic classifications. The Computer Journal **11** (1968)

[57] Joachims, T.: A statistical learning learning model of text classification for support vector machines. 24th annual international ACM SIGIR conference on Research and development in information retrieval (2001)

[58] Joachims, T.: Transductive Inference for Text Classification using Support Vector Machines. 16th International Conference on Machine Learning (ICML-99). Morgan Kaufmann Publishers, San Francisco, US, Bled, Slovenia (1999)

[59] Kanehisa, M., Goto, S.: KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Res. **28** (2000) 27-30

[60] Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M.: RQL: A Declarative Query Language for RDF. 11th International World Wide Web Conference. ACM, Honolulu, Hawaii, USA (2002)

[61] Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM – a Pragmatic Semantic Repository for OWL International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005) at the 6th International Conference on Web Information Systems Engineering (WISE 2005), New York City, NY, USA (2005)

[62] Kleinberg, J.M.: Authoritative Sources in a Hyperlinked Environment. ACM-SIAM Symposium on Discrete Algorithms (1998)

[63] Kochut, K.J., Janik, M.: SPARQLeR: Extended Sparql for Semantic Association Discovery. 4th European Semantic Web Conference (ESWC 2007), Innsbruck, Austria (2007)

[64] Kucera, H., Francis, W.N., Carroll, J.B.: Computational Analysis of Present Day American English. Brown University Press (1967)

[65] Laender, A.H.F., Ribeiro-Neto, B.A., Silva, A.S.d., Teixeira, J.S.: A Brief Survey of Web Data Extraction Tools. SIGMOD Record **31(2)** (2002) 84-93

[66] Leech, G., Garside, R., Bryant, M.: CLAWS4: The tagging of the British National Corpus. 15th International Conference on Computational Linguistics (COLING 94), Kyoto, Japan (1994)

[67] Lenat, D., Guha, R.V.: Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project. Addison-Wesley (1990)

[68] Lewis, D.D., Ringuette, M.: A comparison of two learning algorithms for text categorization. 3rd Annual Symposium on Document Analysis and Information Retrieval (1994)

[69] Lewis, D.D., Yang, Y., Rose, T., Li, F.: RCV1: A New Benchmark Collection for Text Categorization Research. Journal of Machine Learning Research **5** (2004) 361-369

[70] McBride, B.: Jena: Implementing the RDF Model and Syntax Specification. Tenth International World Wide Web Conference: Semantic Web Workshop, Hong Kong (2001)

[71] McCallum, A.K.: Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering.: http://www.cs.cmu.edu/~mccallum/bow (1996)

[72] McCarthy, J.: Notes on Formalizing Contexts. 5th National Conference on Artificial Intelligence, Los Altos, CA, USA (1986)

[73] Mitchell, T.: Machine Learning. McGraw Hill (1997)

[74] Moschitti, A., Basili, R.: Complex Linguistic Features for Text Classification - a comprehensive study. Advances in Information Retrieval, Vol. 2997/2004 (2004)

[75] Nagarajan, M., et al.: Altering Document Term Vectors for Classification - Ontologies as Expectations of Co-occurrence. LSDIS Technical Report (November, 2006)

[76] Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning **39** (2000) 103-134

[77] NLM: Medical Subject Heading (MeSH). The National Library of Medicine, Bethesda, MD

[78] NLM: Unified Medical Language System (UMLS). The National Library of Medicine, Bethesda, MD

[79] Patel-Schneider, P.F.: A Tale of Two Layerings. http://www.w3.org/2001/sw/meetings/tech-200303/all-pfps.htm (March 2003)

[80] Patel-Schneider, P.F., Hayes, P., Horrocks, I.: OWL Web Ontology Language Semantics and Abstract Syntax. http://www.w3.org/TR/owl-semantics/ (10 Feb 2004)

[81] Perry, M.: Test Ontology Generation Tool (TOntoGen). http://lsdis.cs.uga.edu/projects/semdis/tontogen/

[82] Ponzetto, S.P., Strube, M.: Deriving a Large Scale Taxonomy from Wikipedia. Twenty-Second Conference on Artificial Intelligence (AAAI'07), Vancouver, Canada (2007)

[83] Potthast, M., Stein, B., Anderka, M.: A Wikipedia-based Multilingual Retrieval Model. 30th European Confrence on Information Retrieval, Glasgow, Scotland (2008)

[84] Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF - http://www.w3.org/TR/rdf-sparql-query/.

[85] Ramakrishnan, C., Kochut, K.J., Sheth, A.P.: A Framework for Schema-Driven Relationship Discovery from Unstructured text. 5th International Semantic Web Conference (ISWC2006), Athens, GA, USA (2006)

[86] Ramakrishnan, C., Milnor, W.H., Perry, M., Sheth, A.P.: Discovering Informative Connection Subgraphs in Multi-relational Graphs. SIGKDD Explorations **7 (2)** (2005) 56-63

[87] Raptor: http://librdf.org/raptor/.

[88] Rindflesch, T., Tanabe, L., Weinstein, J., Hunter, L.: EDGAR: extraction of drugs, genes and relations from the biomedical literature. Pacific Symposium on Biocomputing (2000)

[89] Rosso, P., Ferretti, E., Jiménez, D., Vidal, V.: Text Categorization and Information Retrieval Using WordNet Senses. 2nd Global WordNet Int. Conf., GWN-2004, Brno, Czech Republic (2004)

[90] Sabidussi, G.: The centrality index of a graph. Psychometrika **31** (1966) 581-603

[91] Sahoo, S.S., Thomas, C., Sheth, A., York, W., Tartir, S.: Knowledge Modeling and its Application in Life Sciences: A Tale of two Ontologies. 15th International World Wide Web Conference (WWW2006), Edinburgh, Scotland (2006)

[92] Schenker, A., Bunke, H., Last, M., Kandel, A.: Graph-Theoretic Techniques for Web Content Mining, Vol. 62. World Scientific, London (2005)

[93] Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. International Conference on New Methods in Language Processing, Manchester, UK (1994)

[94] Schreiber, G., et al.: MultimediaN E-Culture demonstrator. Semantic Web Challenge, 5th International Semantic Web Conference ISWC2006, Athens, GA, USA (2006)

[95] Seaborne, A.: RDQL - A Query Language for RDF. (2004)

[96] Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys (CSUR) **34** (2002) 1 - 47

[97] Semagix: Anti-Money Laundering - CIRAS. http://www.semagix.com/solutions_ciras.html.

[98] SemanticDiscovery: Semantic Discovery: Discovering Complex Relationships in Semantic Web. http://lsdis.cs.uga.edu/Projects/SemDis/.

[99] Shadbolt, N.R., Berners-Lee, T., Hall, W.: The Semantic Web Revisited. IEEE Intelligent Systems **21(3)** (2006) 96-101

[100] Sheth, A.P., et al.: Semantic Association Identification and Knowledge Discovery for National Security Applications. Journal of Database Management (2004)

[101] Sheth, A.P., et al.: Semantic Content Management for Enterprises and the Web. IEEE Internet Computing **July/August 2002** (2002)

[102] Sheth, A.P., Ramakrishnan, C.: Semantic (Web) Technology In Action: Ontology Driven Information Systems for Search, Integration and Analysis. IEEE Data Engineering Bulletin **26 (4)** (2003) 40-48

[103] Sidoroff, T., Hyvonen, E.: Semantic E-government Portals - A Case Study. ISWC-2005 Workshop Semantic Web Case Studies and Best Practices for eBusiness SWCASE05, Galway, Ireland (2005)

[104] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical owl-dl reasoner. Web Semantics: Science, Services and Agents on the World Wide Web **5 (2)** (2007) 51-53

[105] Snoussi, H., Magnin, L., Nie, J.-Y.: Toward an Ontology-based Web Data Extraction. Workshop on Business Agents and the Semantic Web, Calgary, Alberta, Canada (2002)

[106] Strube, M., Ponzetto, S.P.: Wikirelate! Computing Semantic Relatedness Using Wikipedia.: Twenty-First Conference on Artificial Intelligence (AAAI'06), Boston, Massachusetts (2006)

[107] Supekar, K., Patel, C., Lee, Y.: Characterizing Quality of Knowledge on Semantic Web. 15h International Florida Artificial Intelligence Research Society Conference, Pensacola, FL, USA (2002)

[108] Swanson, R.D.: Migraine and Magnesium: Eleven Neglected Connections. Perspectives in Biology and Medicine **31** (1988) 526-557

[109] Tartir, S., Arpinar, I.B., Moore, M., Sheth, A.P., Aleman-Meza, B.: OntoQA: Metric-Based Ontology Quality Analysis. IEEE ICDM 2005 Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources, Houston, TX, USA (2005)

[110] Temkin, J.M., Gilder, M.R.: Extraction of protein interaction information from unstructured text using a context-free grammar. Bioinformatics, Vol. 19. Oxford University Press (2003) 2046-2053

[111] Toutanova, K., Manning, C.D.: Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), Hong Kong (2000)

[112] Tsuruoka, Y., Tsujii, J.i.: Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data. Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005), Vancouver, B.C., Canada (2005) 467-474

[113] Tsuruoka, Y., Tsujii, J.i.: Chunk Parser Revisited. 9th International Workshop on Parsing Technologies (IWPT 2005) (2005)

[114] Ullman, J.D.: Implementation of logical query languages for databases. ACM Transactions on Database Systems (TODS) **10** (1985) 289-321

[115] Vargas-Vera, M., et al.: MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. 13th International Conference on Knowledge Engineering and Management, Siguenza, Spain (2002)

[116] Volkel, M., Krotzsch, M., Vradecic, D., Haller, H., Studer, R.: Semantic Wikipedia. 15th International Conference on World Wide Wed (WWW 2006), Edinburgh, Scotland (2006)

[117] Volker, J., Vrandecic, D., Sure, Y.: Automatic Evaluation of Ontologies (AEON). 4th International Semantic Web Conference (ISWC2005), Galway, Ireland (2005)

[118] Volz, R., Oberle, D., Staab, S., Motik, B.: KAON SERVER - A Semantic Web Management System. 12th International World Wide Web Conference (WWW2003), Budapest, Hungary (2003)

[119] Voss, J.: Collaborative thesaurus tagging the Wikipedia way. ArXiv Computer Science e-prints **cs/0604036** (2006)

[120] Welty, C.: Semantic Web Ontologies. W3C Semantic Web Best Practices WG (2005)

[121] Wilkinson, K., Sayers, C., Kuno, H., Reynolds, D.: Efficient Storage and Retrieval of RDF Graphs in Jena2. First International Workshop on Semantic Web and Databases, Berlin, Germany (2003)

[122] Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques (2nd ed.). Morgan Kaufmann, San Francisco (2005)

[123] Wu, S.-H., Tsai, T.-H., Hsu, W.-L.: Text categorization using automatically acquired domain ontology. 6th international workshop on Information retrieval with Asian languages - Volume 11, Sapporo, Japan (2003)

[124] Yang, Y.: Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1994)

[125] Yang, Y., Liu, X.: A Re-Examination of Text Categorization Methods. 22nd annual international ACM SIGIR conference on Research and development in information retrieval, Berkeley, California, United States (1999)

[126] Zaragoza, H., et al.: Ranking very many typed entities on Wikipedia. 16th ACM Conference on Information and Knowledge Management (CIKM 2007), Lisbon, Portugal (2007)

[127] Zelikovitz, S., Hirsh, H.: Improving Short Text Classification Using Unlabeled Background Knowledge. Seventeenth International Conference on Machine Learning (ICML), Stanford, CA (2000)

[128] Zesch, T., Gurevych, I.: Analysis of the Wikipedia Category Graph for NLP Application. Workshop - TextGraphs-2: Graph-based Methods for Natural Language Processing at NAACL Human Language Technologies Conference, Rochester, New York (2007)