

Federated Ontology Search

Vasco Calais Pedro

CMU-LTI-09-010

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave. Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Jaime Carbonell, Chair
Eric Nyberg
Robert Frederking
Eduard Hovy, Information Sciences Institute

*Submitted in partial fulfillment of the requirements
for the degree Doctor of Philosophy
In Language and Information Technologies*

Copyright © 2009 Vasco Calais Pedro

For my grandmother, Avó Helena, I am sorry I wasn't there

Abstract

An Ontology can be defined as a formal representation of a set of concepts within a domain and the relationships between those concepts. The development of the semantic web initiative is rapidly increasing the number of publicly available ontologies. In such a distributed environment, complex applications often need to handle multiple ontologies in order to provide adequate domain coverage.

Surprisingly, there is a lack of adequate frameworks for enabling the use of multiple ontologies transparently while abstracting the particular ontological structures used by that framework. Given that any ontology represents the views of its author or authors, using multiple ontologies requires us to deal with several significant challenges, some stemming from the nature of knowledge itself, such as cases of polysemy or homography, and some stemming from the structures that we choose to represent such knowledge with.

The focus of this thesis is to explore a set of techniques that will allow us to overcome some of the challenges found when using multiple ontologies, thus making progress in the creation of a functional information access platform for structured sources. In this thesis we try to address the question “How do we integrate and use information contained in a set of heterogeneous ontologies?” This question is becoming crucial as the world gets more connected. As the amount of information grows, so does the amount of resources that encode domain knowledge in different forms. At the same time, as we deal with problems that span a large number of domains, such as search, question answering and semantic analysis, the need for concurrent access to multiple ontologies becomes ever more self-evident.

We model our approach on the general framework proposed in Federated Search and transpose the set of problems to the ontological domain. We start by illustrating the variety of available ontologies and focusing on automatic ontology creation. We proceed to describe the use of proactive ontology selection to guide the ontology selection process at the query level. We then propose a set of operators for ontological search centered on the user’s information needs and expand the basic operator set through composition of basic operators. We address the problem of graph merging within Ontology Search and propose a scoring metric for results that is based in the proposed operator set.

Finally, we present results of using the Federated Ontology Search (FOS) engine in a set of task based evaluations. Our task set is comprised of type checking for question answering, concept coverage, concept disambiguation and content recommendation. We show that in all of the evaluated tasks, the FOS system outperforms the use of individual ontologies.

1	INTRODUCTION	6
1.1	MOTIVATING EXAMPLE	12
1.1.1	<i>Question Answering</i>	12
1.1.2	<i>Example 1</i>	12
1.2	BASIC DEFINITIONS	20
1.2.1	<i>Ontologies and Graphs</i>	20
1.3	CHALLENGES TO BE ADDRESSED.....	22
1.4	SUMMARY OF CONTRIBUTIONS	22
1.5	EVALUATION OF RESULTS	23
1.6	OVERVIEW OF THESIS ORGANIZATION.....	24
1.7	SUMMARY.....	24
2	BACKGROUND AND RELATED WORK	26
2.1	ONTOLOGY SELECTION.....	26
2.2	WORD SENSE DISAMBIGUATION.....	27
2.3	RESULT MAPPING AND MERGING	29
3	AUTOMATIC ONTOLOGY CREATION	32
3.1	AUTOMATIC CREATION OF ONTOLOGIES FROM SEMI STRUCTURED RESOURCES.....	32
3.2	RELATED WORK	33
3.3	THE WIKIPEDIA STRUCTURE.....	34
3.4	GENERAL METHODOLOGY	35
3.5	FEATURE EXTRACTION	35
3.6	GENERATING LABELED DATA	36
3.6.1	<i>List Based Labeling</i>	37
3.6.2	<i>Context Based Labeling</i>	37
3.7	DIRECTIONAL FEEDBACK EDGE LABELING ALGORITHM	39
3.8	UMLS AND OKINET.....	42
3.9	EXPERIMENTAL SETTING	43
3.10	RESULTS	43
4	ONTOLOGY SELECTION.....	47
4.1	CHARACTERIZATION OF ONTOLOGICAL RESOURCES	47
4.2	ONTOLOGICAL REPRESENTATION LANGUAGES	52
4.2.1	<i>Knowledge Interchange Format (KIF)</i>	52
4.2.2	<i>Resource Description Framework (RDF)</i>	52
4.2.3	<i>Web Ontology Language (OWL)</i>	53
4.2.4	<i>CycL</i>	54
4.3	PROACTIVE ONTOLOGY SELECTION.....	55
4.3.1	<i>Cooperative Ontology Selection</i>	55
4.3.2	<i>Uncooperative Ontology Selection</i>	56
5	ONTOLOGICAL SEARCH	62
5.1	ONTOLOGY QUERY LANGUAGES	62
5.1.1	<i>Simple Protocol and RDF Query Language (SPARQL)</i>	62
5.1.2	<i>Metaweb Query Language (MQL)</i>	63
5.1.3	<i>User Centric Approach to Ontology Query Languages</i>	64
5.2	PROPOSED QUERY APPROACH	67
5.2.1	<i>Operators</i>	68
5.2.2	<i>Query</i>	80
5.2.3	<i>Query Results</i>	81
5.3	SUMMARY.....	82
6	RESULT MERGING AND SCORING	83
6.1	PROBLEM DEFINITION.....	83

6.2	SEMANTIC DISTINCTION OF RESULTS	84
6.3	CONCEPT NORMALIZATION USING SYNONYM RESOURCES	85
6.4	EDGE NORMALIZATION	88
6.4.1	<i>Label Synonym Identification</i>	88
6.5	MERGING PROCEDURE.....	89
6.5.1	<i>Graph Similarity</i>	89
6.5.2	<i>Granularity Resolution</i>	92
6.6	SCALABILITY	97
6.7	RESULT SCORING.....	98
7	RESULT DISCUSSION	102
7.1	TYPE CHECKING	103
7.1.1	<i>Experimental Setup</i>	103
7.1.2	<i>Results and Analysis</i>	103
7.2	CONTENT MATCHING	106
7.2.1	<i>Task Definition</i>	106
7.2.2	<i>Experimental Setup</i>	106
8	RESULT ANALYSIS	111
9	CONCLUSIONS AND FUTURE WORK	121
9.1	SUMMARY OF CONTRIBUTIONS.....	121
9.2	FUTURE WORK	122
9.3	CONCLUSION	123
10	BIBLIOGRAPHY	127

Table of Figures

Figure 1 - an example of a basic ontology	21
Figure 2 - A graph structure.....	21
Figure 3 - Ontology Extraction Example.....	39
Figure 4 - Example of Directional Labeling.....	40
Figure 5 - Sample Ontology Categorization	49
Figure 6 - RDF Example in XML notation.....	53
Figure 7 - OWL Example	54
Figure 8 - CycL Example.....	55
Figure 9 - Overview of query Run Time ontology Selection	59
Figure 10 - SPARQL Example	63
Figure 11 - MQL Example.....	63
Figure 12 - MQL Result Example	64
Figure 13 - SPARQL Impossible Example.....	65
Figure 14 - MQL Transitive Query.....	66
Figure 15 - Simple relation Query	66
Figure 16 - Search Operator Example	71
Figure 17 - relation operator general form.....	72
Figure 18 – relation operator.....	73
Figure 19 - Concurrent Operator General Form	74
Figure 20 - Concurrent Operator expressed in terms of the relation operator	75
Figure 21 - The define operator as a formulation of the relation operator	76
Figure 22 - Define Operator.....	76
Figure 23 – Base case for children operator	77
Figure 24 - The Children operator as a function of the relation operator	77
Figure 25 – Base case for parents operator	78
Figure 26 - The parents operator as a function of the relation operator	78
Figure 27 - The and operator	78
Figure 28 – and operator	79
Figure 29 - The or operator	79
Figure 30 – or operator	80
Figure 31 - Decomposition of operators	80
Figure 32 - Scenarios for Result Merging.....	84
Figure 33 - Example of Concept Normalization.....	86
Figure 34 - Counter Example of Concept Normalization.....	87
Figure 35 – Pseudo code for concept normalization.....	88
Figure 36 – Localized Boosting Algorithm	91
Figure 37 - Granularity Problem Example.....	93
Figure 38 - Intial Merging.....	94
Figure 39 - Merging Result.....	96
Figure 40 - Result scoring example 1	98
Figure 41 - Result scoring example 2	98
Figure 42 - Result scoring example 3	98
Figure 43 - Example of Mechanical Turk task	107

Figure 44 - Agreement between reviewers	109
Figure 45 - relevancy of products	110
Figure 46 - Agreement between reviewers	110
Figure 47 - Coverage Results.....	113
Figure 48 - Disambiguation Example	116
Figure 49 - Disambiguation Results	117
Figure 50 - precision results.....	119
Figure 51 - Examples of semantically related non-overlapping words	120

Acknowledgements

It is expected that during such a journey one should somehow change, grow as a person. If I were to stand at the doorstep of graduate school, I would have never guessed how much that could be possible. In knowing that the voyage is really just beginning, I hope that, besides bringing me back to CMU once in a while, the rest of it will be at least as amazing as it has been so far.

I would like to start by thanking my advisors, Jaime Carbonell and Eric Nyberg, who made sure that the excitement of being lost always led to good port. To Jaime I thank the many hours of guidance and the support in every part of my stay at CMU. To Eric I thank the instilled respect for sound software principles, learning the value of integration and the power of research in the real world. To a native engineer like me, My advisors were truly an inspiration of the potential that lies in the application of great ideas to real problems. I would also like to thank my committee, Robert Frederking and Eduard Hovy for all the generous comments and support during my thesis which helped me to push myself a little further.

The list of friends grows too numerous, and the number of teams and intramurals, activities and clubs, which made so much of my academic life, would take the best part of a page to acknowledge. Thanks to Ulas, Kenji, Alicia, Kevin, Thomas, Ashish, Betty, Le, Jasmine, Udhai, Sourish, Lucian, Andre, Guy, Catherine, Kornel, Paul (Bennett and Ogilvie), John, Arthur, Yan, Giovanni and so many others that made my life here so much, much better. To the Portuguese speaking, Bernardo, Margarida, Graça, Francisco, André, Sara, Evandro and Rosemeri, thank you for bringing a little bit of home to Pittsburgh. To all of the people that I have the honor to call friends, thank you.

To my mother Emilia Ribeiro, the guiding inspiration in my career, I thank from the bottom of my heart for always demanding more, for the sacrifices made to ensure a better life for her children and for the unwavering love and support in everything I do.

To my grandmother Maria Helena Santos, who died before seeing the end of this journey, I hope that she can see in my soul the deep love and complete gratitude that I feel for having had her in my life. She was my rock, my shelter and her complete selflessness left a mark in all who were touched by it.

To Joao and Célia Carreira, my in-laws, avid supporters of everything CMU, I thank for the support and wonderful energy that every visit would bring. To my sister Catarina Pedro, my father Manuel Pedro, and the rest of my family, I thank you for the support and confidence in my work.

To my daughters Beatriz, Mariana and Sofia Calais-Pedro, I thank you for putting everything into perspective. Your innocent eyes make everything simple; your smiling faces make everything wonderful.

Finally, I want to thank the love of my life, my wife, my friend and companion of so many travels, Maria João Calais Pedro. Her sacrifice in face of adversity, her tenderness on tired days, her humor on the face of desperation, her amazing management skills with lack of resources, her willingness to be by my side in every turn and her enduring and ever present love have really made this thesis possible. I will always love the pilgrim soul in you.

1 Introduction

In everyday life we use structured and semi-structured information collections. They play a fundamental role in providing information that we can use to fulfill daily tasks such as purchasing decisions, map routes and voting decisions. Yet, when it comes to computer applications that use structured domain information, most applications use only one information source, thus limiting themselves severely in the breadth and amount of supporting information used within the goal of that application. The main reason for this is the lack of a transparent and efficient way to access, process and utilize a set of structured domain information sources. Given that any encoded representation of domain knowledge represents the views of its author or authors, using multiple sources require us to deal with several significant challenges, some stemming from the nature of information itself, such as polysemy and structural ambiguity, and some stemming from the structures that we choose to represent the collective data, such as graph merging.

The focus of this thesis is to explore a set of techniques that will allow us to overcome some of these challenges, thus making progress in the creation of a functional information access platform for structured sources. In this thesis we try to address the question “How do we integrate and use information contained in a set of heterogeneous resources that model domain knowledge?”

This question is becoming crucial as the world gets more connected. As the amount of information grows, so does the amount of resources that encode domain knowledge in different forms. At the same time, as we deal with problems that span a large number of domains, such as search, question answering and semantic analysis, the need to use domain knowledge becomes ever more self-evident. The problem is that, unlike the expert systems of the 70’s which focused on a small number of domains, the current problems require a global solution to the domain knowledge access problem.

Within computer science, the resources that encode world and domain knowledge are called taxonomies, dictionaries, knowledge bases, expert systems or more generally

Ontologies. Each name represents a slight variation on the properties of the domain resource, but as usual, the number of definitions far outnumbers the things being defined. As a first step towards a general approach we consider those domain resources to be a type of ontology, which we now define.

Despite several definitions (Guarino 1998), an Ontology can be defined as a model that represents a domain and is used to reason about objects in that domain and the relations between them. It is usually composed of concepts, relations between those concepts, concept properties and instances. Within the scope of this work we consider taxonomies, semantic nets (Quillian 1967) and lexical resources such as Wordnet (Miller 1995) as ontologies. The use of ontologies is now widespread in areas as diverse as biomedical research, information extraction and knowledge engineering and management.

For the purposes of our research, an ontology can be as simple as a semantic network (Quillian 1967), where no distinction is made between concepts and instances, and the only relation possible is of the *is-a* type, or as complex as CYC (Lenat 1995), with a clear distinction between concepts and instances, where multiple inheritance is allowed and there is an extremely rich set of possible relations. In order to accommodate the breadth of representational possibilities we reduce the representation to the general form of graph, where the nodes represent concepts and the edges represent relations, making no assumption as to the limit of what relations can exist. This casts the majority of the addressed problem as a graph approach problem, addressing ontology merging as a form of graph merging.

In the last decades the number of available ontologies has grown considerably. Several proprietary and open-domain ontological resources such as CYC (Lenat 1995), SUMO (Niles and Pease 2001), Omega (Philpot, Fleischman et al. 2003), Scone (Fahlman 2005), ThoughtTreasure (Mueller 1997), Wordnet (Miller 1995), VerbNet (Schuler 2003), Framenet (Baker, Fillmore et al. 1998) and Propbank (Kingsbury and Palmer 2002) have become available. Swoogle (Ding, Finin et al. 2004) has now indexed more than 10 000 ontologies. These resources offer the promise of easily-accessible, open-domain

ontological information, but the existence of such diverse ontologies raises the issue of information merging and reuse. A comparison of the ontologies reveals both redundant and complementary coverage, but the variety of frameworks and languages used for ontology development makes it a challenge to merge query results from different ontologies. The number of available languages for ontological knowledge engineering such as RDF, OWL, DAML+OIL and CYCL, combined with the existence of independent interfaces aggravates the issue.

The problem is not just seen in general domains, where one would expect the knowledge to be more readily available, but also in more specific domains such as the medical domain. In the medical domain the number of ontologies has also grown considerably. These ontologies enable the use of previous medical knowledge in a structured way. Applications of medical ontologies include: more effective search of patient records, hospital quality improvement programs, (semi)automatic ICD-9 coding for insurance reimbursement, preliminary symptom-based diagnosis, ambiguity reduction when choosing medical tests, and classification of diseases, symptoms, and other medical concepts. For example, when trying to answer whether a patient was prescribed *Aspirin* (for hospital quality improvement measures), one needs to consider similar terms (such as *Ecotrin*, *Bayer pain reliever*, etc). Also, when performing (semi)automatic patient ICD-9 coding, it is useful to map conditions that can be described in various ways (*Heart Attack* can be also stated as *AMI* or *MI* or *Myocardial Infarction* or simply *Infarction*). For preliminary diagnosis at the point of care, ontologies can help by quickly returning diseases that have a given set of symptoms (instances of symptoms and diseases are concepts related by the “symptom of” relationship).

As an effort to solve some of the problems mentioned previously, several proprietary and public ontological resources, either in the form of the ontology data or as a public web service, such as MESH (Lipscomb 2000) and SNOMED (Spackman, Campbell et al. 1997) have become available. UMLS (Bodenreider) is a resource that combines a number of other resources, is rapidly becoming a de facto standard for medical ontologies, containing more than 100 dictionaries. Other medical ontologies include: RadLex

(Radiology Information Resource), OMIM (Online Mendelian Inheritance in Man), MEDCIN (medical terminology), LOINC (Logical Observation Identifiers Names and Codes) and ICD-9/ICD-10 Codes.

The existence of the integration problems across both general and specific domains suggest an urgent need for frameworks and techniques that allow for the effective integration of the information contained in the different sources that pertain to these domains. The necessity of using more than ontology can be illustrated by practical scenarios such as Question Answering, Content Recommendation and Sentiment Analysis. In all of the open-domain problems, it is unlikely that one single ontology will provide the required coverage for concept understanding and content correlation.

In Question answering, there is the need to verify that potential answers are of the expected answer type. For example, given the question “what is the capital of Italy?” we are expecting a city to be the answer. Thus every candidate answer should at least be of the type city. While it is possible to have one ontology that covers all the cities in the world, that is but one single type of question, if we take multiple type of questions we quickly realize that in order to adequately check potential answers we will need to combine several ontologies.

In content recommendation, we need to first identify the concepts that relate to both the content that we will base our recommendation as well as the content that we might recommend. Again in this scenario in order to correctly identify the concepts of interest, as well as correlate them across content, we will most likely require the use of several ontologies. By and large, the inexistence of general purpose framework significantly hampers the development of solutions that rely on these ontologies.

Currently, the main approaches to a solution for these problems focus on ontology integration, by creating a mapping between the concepts and relations of different ontologies. Some cases, such as the Semantic Web project (Bemers-Lee, Hendler et al. 2001) primarily rely on merging two ontologies by establishing a full mapping between

them. Some efforts have tried to produce a merged ontology automatically using a bottom-up approach such as FCA-Merge (Stumme and Maedche 2001); most involve some degree of semi-supervised mapping. Other approaches, such as the one taken by CYC, try to absorb other ontologies into a single main ontology while maintaining coherence (Reed and Lenat 2002). One disadvantage of these approaches is the prohibitive cost of producing a mapping or absorbing an ontology, given their increasing scale and rate of availability. Another disadvantage is that it is not always possible to establish a one-to-one mapping between the concepts and relations in one ontology and the concepts and relations in another. Furthermore, there is the problem of keeping the mappings updated as the original ontologies evolve. A large number of available ontologies are considered works in progress and are updated frequently, which implies a constant updating of any mappings associated with those resources.

The aforementioned approaches have several drawbacks, as discussed in (Klein 2001; Serafini and Tamilin 2005), such as (i) non-scalability, (ii) loss of language and reasoning specificity of distinct ontologies, (iii) loss of privacy and autonomy of ontological knowledge (iv) language level mismatches such as syntax mismatches, differences in logical representation and different semantic primitives and (v) Ontology level mismatches, such as difference in scope, coverage and granularity, making this challenge thus far too daunting in practice. A second approach is to query more than one ontology via different interfaces, and interpret the results of each ontology individually, essentially moving the entire challenge from the ontology provider to the application builder.

We propose to build an ontological middleware level which supports retrieval and merging of small fragments of ontologies which permits us to:

- Query multiple ontologies and then merge the query results from multiple knowledge base systems, much like Federated Search in information retrieval (Si and Callan 2005).

- Follow ontological chains and inferences across ontologies, using partial query results from one ontology to query another. This is a more complex version of cross-data-base joins, where the data schemas are sufficiently compatible.

The majority of applications that use ontological information would benefit from an approach that models the information need, queries the relevant ontologies and retrieves the best result while providing a single unified interface to the client application. If we look to other domains for inspiration on how to proceed, we can find a similar problem in the field of Federated Search (Callan 2000; Fryer 2004). Information Retrieval is usually based on a single database model of text retrieval. But to cope with proprietary information spread around the world in separate databases, distributed information retrieval explicitly models multiple databases for text retrieval. Each database is queried independently, the results are merged when possible and a new global ranking is established.

In the same fashion, we can model our ontologies as individual sources, construct a query that describes the information need, query each ontology independently and merge the results into one ranked list.

Using Federated Ontology Search we can parallelize query execution while respecting the structure of the individual ontologies, taking advantage of both redundant and complementary knowledge in the available ontologies to improve the overall performance of the system.

The hypothesis governing this thesis is as follows; by using a federated approach we hope to be able to overcome the problem of ontology integration and reuse that prevent the use of multiple ontologies within open domain applications. This thesis will defend and explain the advantages of incorporating the set of available ontologies using a federated approach and decompose the problem into four sub problems, namely: Resource selection, Query Execution, Result Merging and Ranking.

1.1 *Motivating Example*

Although there are many applicable areas for this research, type checking in the factoid QA domain is suitable to prove the utility of our approach. The advantages of this are threefold. First, the application of this research in factoid QA can be well defined and the ontological operations involved are conceptually clear but yet not trivial. Second, the training and test data created for the TREC QA evaluations (Voorhees 2003), consisting of corpora, question sets and answers keys, provide the required evaluation material.

1.1.1 *Question Answering*

In Question Answering, the goal is to take a question in natural language and provide an answer also in natural language. In the JAVELIN I question answering system (Nyberg, Mitamura et al. 2003) a set of modules is used, specifically the Question Analyzer (QA module) module, which analyzes the question; the Retrieval Strategist (RS) modules, which retrieves the relevant documents; the Information Extractor (IX) module, which analyzes the documents retrieved by the RS and provides candidate answers; and finally the Answer Generator (AG) module which analyzes the candidate answers and generates a final ranked list of answers. Ontological information is typically used within JAVELIN to determine the relation between the expected answer type and the candidate answers or to provide answer verification, if the answer can be found directly in an available ontological resource (Ko, Hiyakumoto et al. 2006).

1.1.2 *Example 1*

For this example let's assume that we have access to the following ontologies described in Table 1.

Ontology
CYC

Wordnet
U.S. Gazetteer

Table 1 - Available Ontologies for example 1

Let's consider the question: *What is the largest city in Germany?*

Part of the QA module's responsibilities in analyzing the question is to determine, amongst other things, the type of answer we are expecting and the constraints on that answer. In this case the QA module determined that the type of answer is *location* with an additional constraint that the answer must be a *city*. Table 2 shows a partial output from the QA and Table 3 shows Javelin's output.

Question	What is the largest city in Germany
Answer Type	<i>location</i>
Subtype Constraint	<i>city</i>

Table 2 – Output of the Question Analyzer. We can see the constraints expected in the answer.

AG output		
Rank	Answer	Judgment
1	Italy	Not a City
2	Berlin (Correct)	Correct
3	Horten	Incorrect
4	Norway	Not a City
5	South Africa	Not a City
6	Dusseldorf	Incorrect
7	Spain	Not a City
8	Moscow	Incorrect
9	France	Not a City

10	Swiss	Not a City
11	London	Incorrect
12	Oslo	Incorrect
13	Cologne	Incorrect
14	Pretoria	Incorrect

Table 3 – Output of the Answer Generator

As we can see the correct answer is ranked second. Furthermore, the first ranked answer, as well as several others, violates the constraints set by the QA module because it is not a city. We need a way to enforce those constraints on the candidate answers. One possibility is to use ontological knowledge to verify these constraints, but given that QA is an open domain area, one single ontology most likely will not provide adequate coverage for the type of interest.

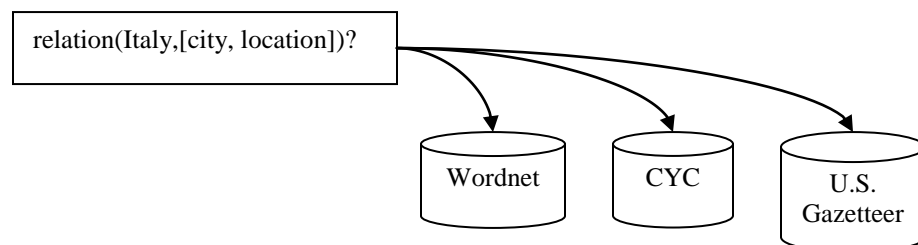
Consider an example which illustrates the federated ontology search approach. The main idea in this case is that we would submit each of the candidate answers along with the answer type constraints for verification.

I will show the procedure for the first two answers in the ranked set. The rest of the answers would proceed in similar fashion.

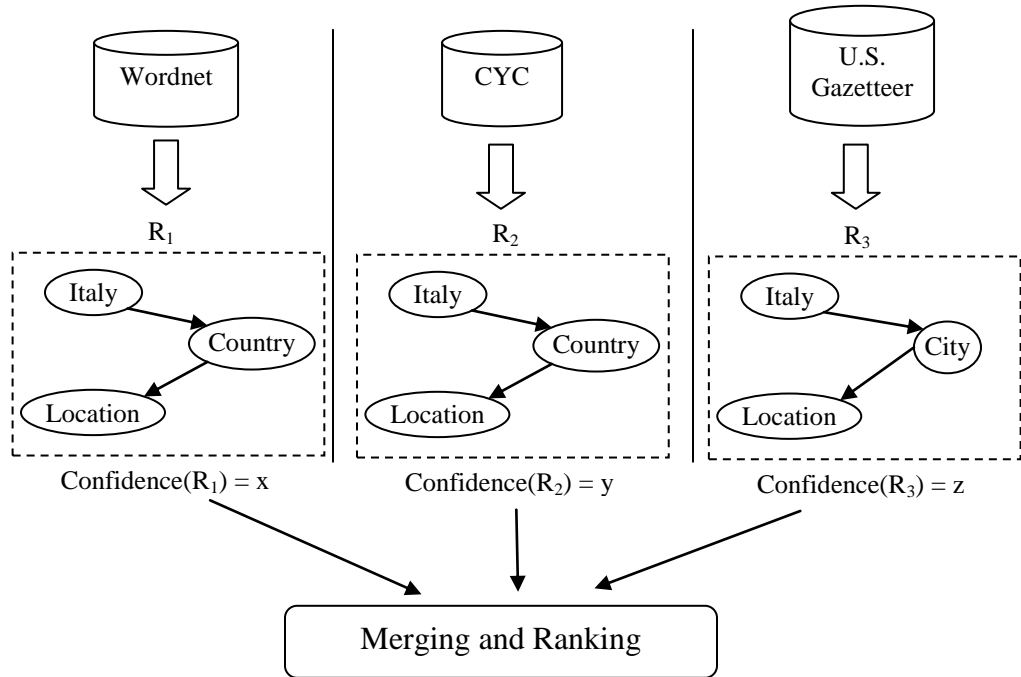
Constraints : *city, location*

a) Answer1 : *Italy*

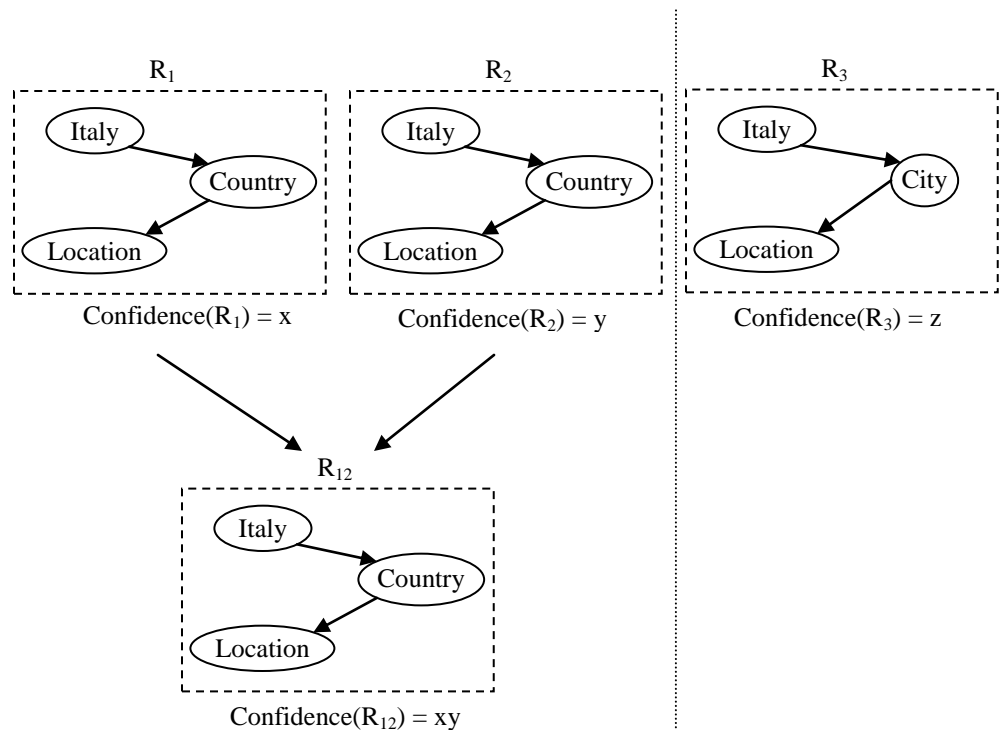
a. *The available ontologies are queried regarding if there is any relation between Italy and city*



- b. *The results are collected. Each result contains a confidence of the correctness of the result as well as a source confidence. Note that in the case of the U.S Gazetteer, Italy is a city in Texas.*



- c. *Merge and rank the results. When two results are merged, their confidence is boosted*



- d. Finally a ranked list is produced by the server and the highest ranking answer is that Italy is a country rather than a city.

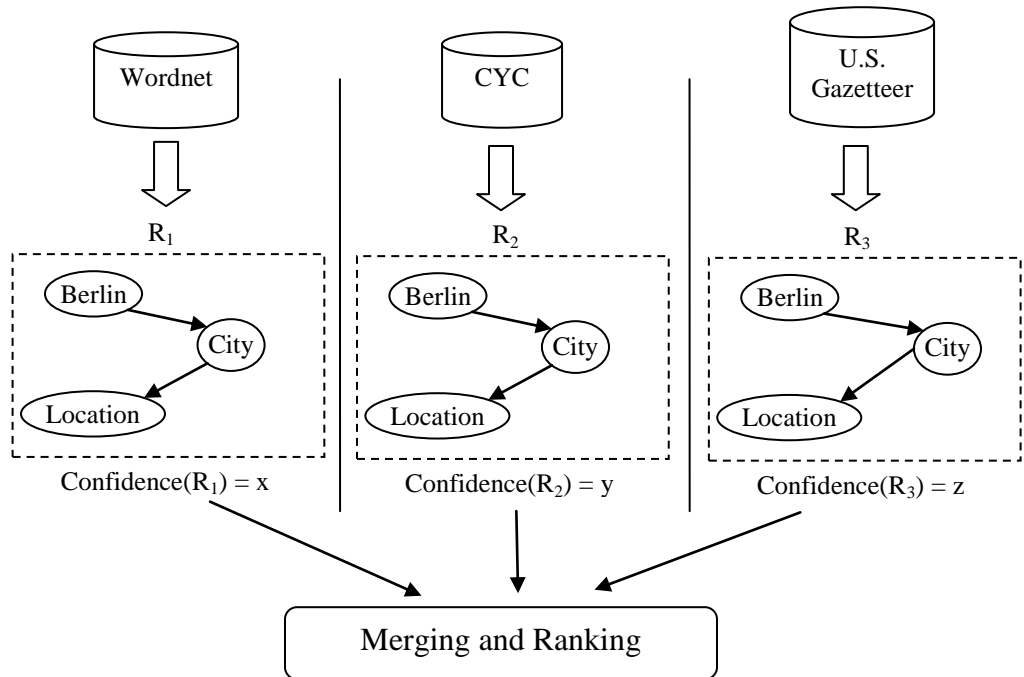
Rank	Result
1	<p>Confidence(R_{12}) = $f(x,y)$</p>
2	<p>Confidence(R_3) = z</p>

Where the confidence of R_{12} is given by a function f that combines x and y .
also, since the ranking is based on the confidence, $f(x,y) > z$.

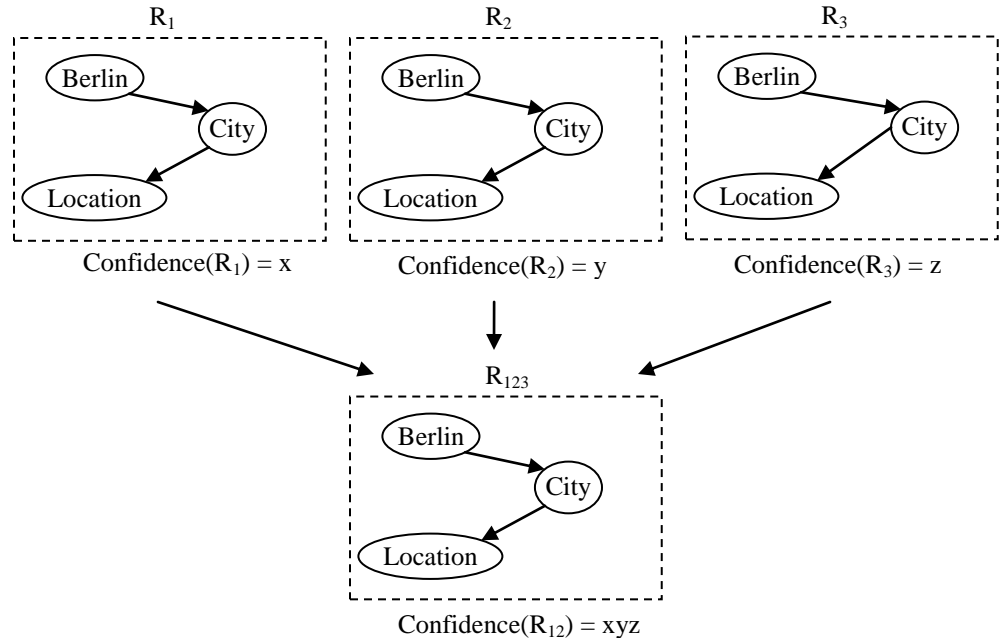
b) Answer 2: *Berlin*

a. *Step a would be the same as before*

b. *The results are collected. Each result contains a confidence of the correctness of the result as well as a source confidence*



c. *Merge and rank the results. When two results are merged, their confidence is boosted.*



d. Finally a ranked list is produced by the server shows Berlin as the only answer

Rank	Result
1	$\text{Confidence}(R_{123}) = xyz$

After applying this procedure to all the answers, Table 4 shows the final ranked list of answers. We can see that the correct answer is now on ranked first.

AG output			
Previous Rank	New Rank	Answer	Judgment
2	1	Berlin (Correct)	Correct
3	2	Horten	Incorrect
6	3	Dusseldorf	Incorrect
8	4	Moscow	Incorrect
11	5	London	Incorrect
12	6	Oslo	Incorrect
13	7	Cologne	Incorrect
14	8	Pretoria	Incorrect

Table 4 – Output of the Answer Generator

1.2 Basic Definitions

1.2.1 Ontologies and Graphs

Although there is no consensual definition of *ontology*, a good start comes from G. Stumme and A. Maedche (Stumme and Maedche, 2001). The authors claim that most ontologies share a few common items such as

- Concepts, a hierarchical *IS-A* type relation and further relations.
- Some ontologies have constraints, functions or axioms

A basic ontology definition could given by a tuple $O := (C; is\ a; R)$, where C is a set whose elements are called concepts, *is a* establishes a partial order on C and R is a set whose elements are called relation names. An example is given below.

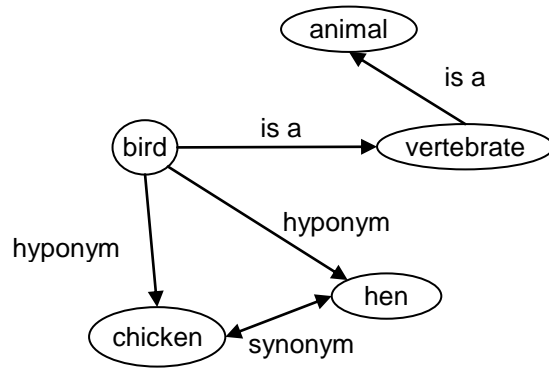


Figure 1 - an example of a basic ontology

A graph definition could be given by $G = (V,E)$ where V is the set of vertices and E is the set of edges. An example is given below.

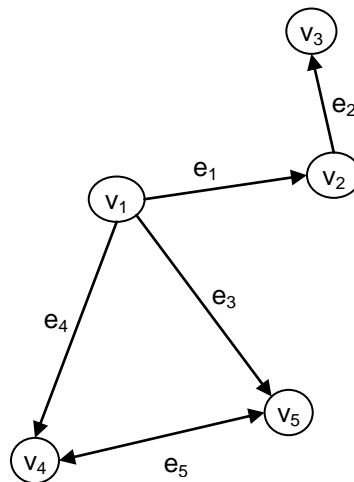


Figure 2 - A graph structure

Given the two definitions one can see that graphs represent the basic structure of ontologies very well. Vertices are considered concepts, Labeled edges as relations.

Query is a request for information from the set of existing ontologies. It is comprised of operators, as defined in section 5.2.1.

Result is the rooted directed acyclic graph (RDAG) that results from executing a query.

1.3 Challenges to be addressed

In creating a federated approach to ontology querying, several challenges must be addressed and will shape the solutions we will develop.

Federated Search identifies three key areas of research for a problem solution. We will show that the same problems apply in the area of Federated Ontology Search:

Resource selection focuses on the problem of selecting the correct ontologies from within the available ontology set. In order to increase efficiency and reduce ambiguity, the selected ontology set will provide the range of possible answers to the query as well as corroborating evidence for the answers provided.

Query Execution will focus the actual querying mechanism and the set of operations required to provide a query language that is both powerful and fine grained enough to provide the desired functionality.

Result merging and scoring will focus on the challenges of taking each individual result and merging the results that corroborate each other. This is one of the central topics addressed by this thesis. Scoring refers to ranking the results in terms of the amount of information they are providing and the relevance of that information.

1.4 Summary of contributions

The main contributions of this thesis are:

A framework to incorporate, query and locally merge the knowledge contained in a set of ontologies. This framework addresses the issues of ontology selection, ontology querying and result merging and scoring.

A system that implements the proposed ideas, creating a platform available for research that incorporates a number of available resources with a unique and transparent interface, running as a web service.

An evaluation of this system to two different problem areas, namely type checking and content recommendation. The evaluation will follow a task-based evaluation methodology.

A test set that can be used by other approaches to evaluate and develop further the methodologies proposed in this research. This test set will be made public and will contain labeled data as well as the results obtained in this thesis, as an improved baseline.

1.5 Evaluation of results

Given the subjective nature of ontology evaluation we will demonstrate the results of this thesis using a task based evaluation where we will compare using the proposed federated approach versus a baseline of the individual ontologies, when used separately. We will do so within two different tasks, Type Checking within question answering and Content Recommendation. Content Recommendation is in turn composed of several sub tasks, specifically; **String Coverage**, where we measure the coverage provided by the Federated Approach; **Concept Co-Disambiguation**, where we measure the ability to disambiguate concepts using the Federated Approach; and finally **Concept Set Matching**, where we measure the ability to match sets of concepts using the federated approach. These tasks will allow us to observe the impact of using a federated approach in coverage and precision within the main task of content recommendation. Using FOS, we demonstrate a performance increase of 30% using the F1 measure. on type checking better than both the individual baselines and the baseline formed by summing the individual baselines. Also, Using FOS, we demonstrate an increase in precision of 47% on Content Recommendation when compared to the baselines obtained by using the individual ontologies as well as the baseline obtained by combining the individual baselines.

1.6 Overview of thesis organization

This document is organized as follows. In the rest of this chapter we will give a motivating example, define the basic principles of ontologies and graphs and provide an overview of the federated approach to ontology querying. In Chapter 2 we will talk about the related and background work and how it relates to the current work. In Chapter 3 we will describe innovations on the automatic process of ontology creation. We will then talk about ontology selection in Chapter 4, followed by ontological search in chapter 5 and we will conclude the presentation of the framework in chapter 6 with discussion result merging and scoring. In Chapter 7 we will present the results of the experiments and the discussion of those results and finally in Chapter 8 we will conclude with final remarks and future work.

1.7 Summary

In this chapter we explained and motivated the goal of this thesis, which is to create a framework for ontology integration and search. We described the main problems with the current approaches and provided an hypothesis for the solution of the challenges that the creation of such a framework presents.

To achieve our goal with these challenges in mind we use the idea of federated search and transport the sub problems found in federated search to the domain of federated ontology search. Our approach assumes independence of ontological resources and provides an elegant way to incorporate new resources without having to merge them in their entirety. Rather we focus on merging the results of small operations on each ontology, represented in graphs.

We give a motivating example and describe the fundamental definitions that will be used throughout this document. Finally we describe the expected evaluation metrics and baseline and layout the organization of this thesis document.

We believe that this thesis represents a significant improvement over the current methods of ontological integration by creating an extensible framework that is representation and location independent while providing the parallelism required for efficient scalability.

2 Background and Related work

The work described on this thesis is orthogonal to many areas, such as Information Extraction, Information Retrieval, Ontology Management, Graph Merging, Federated Databases and Semantic Analysis, making it impractical to adequately cover and explain all the correlations between the aforementioned areas and Federated Ontology Search. Therefore we will limit ourselves to works that have a direct comparison with the work described in this thesis, either as a whole or in specific parts. More specifically we will look at works that have an impact in the two main areas of this thesis, namely ontology selection and result merging.

2.1 *Ontology Selection*

Ontology selection deals with the selection of an ontology given a query. SWOOGLE (Ding, Finin et al. 2004) uses traditional Information Retrieval techniques to retrieve semantic web documents (SWD), specifically character based N-Grams, n-character segments of the text which spans inter-word boundaries, or URIs as keywords. The system indexes ontologies primarily designed with the OWL language which supplies by design a set of metadata which is extremely useful for identification of the SWD. Since the words are usually compounded into URI terms, this N-Gram approach is particularly efficient to index and retrieve the SWD. While this approach provides an efficient index for the acquired ontologies, each ontology is considered a totally independent resource. SWOOGLE lacks the ability to merge the acquired ontologies, either globally or in part, representing a catalog of ontologies rather a unified ontological resource. The ontology selection returns all the ontologies that contain the required term, much like retrieving documents, providing little understanding of the different grouping of each retrieved ontology in respect to different representations of concepts.

Link analysis is used in (Patel, Supekar et al. 2003; Zhang, Vasconcelos et al. 2004) to rank ontologies in respect to queries in the OntoSearch system and in the OntoKhoj system. Alani and Brewster in (Alani and Brewster 2005) create the AKTIVERANK algorithm, aggregate a number of measures that look into the structural features of

concepts such as concept similarity and structural density. Both these works allow for the ranking of ontologies as a independent resources, but they do not provide rankings for individual query results, that is, they do not segment the particular portion of the ontology that answers the performed query, thus not allowing for the effective merging of the ontology subsets. This is discussed in Result Merging and Scoring

It is important to note that this work, despite being inspired by the framework of Federated Search, it is not a direct extension of that framework. Federated search {Fryer, 2004 #157},{Si, 2005 #29},{Lu, 2006 #159}. While in federated search the goal is to search full text documents, typically indexed by keywords, Federated Ontology Search deals with searching over large graphs and combining sub graphs. The metaphor works in so far as the goal of selecting resources, but not in the type of information of the algorithms required for combination. In fact, in Federated Search the combined ranking does not use merging of individual results.

2.2 Word Sense Disambiguation

Necessary to the task of ontology selection is the subtask or word sense disambiguation. Word Sense Disambiguation is the problem of determining the actual meaning of a word given a specific context. This task has been studied extensively and while it is not the main focus of this thesis, it is important to note the most prominent works in the area.

Within the context of this work word sense disambiguation can be cast as mapping the strings in the query to concepts in the ontologies. The area of word sense disambiguation is described in detail, along with algorithms and applications in the book “Word Sense Disambiguation, Algorithms and Applications” (Agirre and Edmonds). Within word sense disambiguation there are three types of approaches, namely Knowledge-based, Unsupervised corpus-based and supervised corpus-based.

Knowledge based refers to using a hand crafted set of disambiguation rules or a set of heuristics to decide the correct sense of the word, the seminal work of Michael Lesk {Lesk, 1986 #153}, estimates the word sense by overlapping the definition of the

ambiguous words in dictionaries with nearby words in the text. **Unsupervised corpus-based** methods cluster word occurrences in text, automatically inducing senses. The intuition here is that similar word meanings occur in similar contexts, so by clustering the occurrences of words in text, we can induce the different meanings. These methods are typically hard to evaluate since the induced meanings still have to be mapped to a known dictionary. A well known algorithm of this family is given in Gale {Gale, 1992 #155}. **Supervised corpus-based** techniques use machine learning techniques such as SVM applied to manually tagged corpora to deduce the word sense, a prominent example is given in the work by Eric Brill and Jun Wu {Brill, 1998 #156}.

The techniques described in this thesis focus on using graph approaches to word sense disambiguation, relying on the structure of the relations that connect concepts to deduce the correct sense. They are more a form of unsupervised ontology based sense disambiguation.

Resnik in (Resnik 1995) uses information content, as defined in (Ross 1976), to determine the semantic similarity of two concepts, the author restricts himself to the use of is-a relations to calculate the concept similarity. While this is extremely useful, it does not provide a complete solution for ontologies that contain more than is-a type relationships, since only is-a relations are considered. One could think of extending the relationship set used in determining the similarity measure to include other relationships, but the characteristics of each relation type differ thus making it impossible to extend the algorithm easily. Jiang and Conrath (Jiang and Conrath 1997) combines a lexical taxonomy with corpus statistical information to measure semantic similarity between words and concepts. The technique suggests promising results for local domain ontologies, but the lack of coverage of the lexical taxonomy, as well as the need to have common corpora for the available ontologies reduces the suitability of the technique for the application to an open ended set of ontologies.

2.3 Result Mapping and Merging

Although work as been done in ontology integration such as (Reed and Lenat 2002) and (Hovy, Fleischman et al. 2003), where the goal is to incorporate several ontologies into one larger ontology, recently the focus seems to be in ontology mapping. In our perspective, incorporating several ontologies into one comprehensive resource inevitably leads to integration problems. Hovy and Fleischman's work on Omega suggests that there is a threshold above which every new concept being added causes an increasing number of collisions with existing concepts, making it impracticable to merge a large set of ontologies.

Stumme and Maedche in (Stumme and Maedche 2001) based their work on the work of Ganter and Wille's (Ganter and Wille 1997) work on formal concept analysis. Their method, the FCA-Merge is semi-automatic method for merging ontologies that uses natural language techniques to derive a lattice of concepts which is then explored by a knowledge engineer. The FCA-Merge assumes that a corpus relevant to both ontologies to be merged is available and relies on the availability of classified instances in those ontologies. The assumption of the existence of a corpus common to both ontologies reduces the usability of this method for a large set of ontologies, despite a generally high level of accuracy in the experiments performed.

Using the Barwise-Seligman theory of information flow (Barwise and Seligman 1997), Kalfoglou and Shorlemmer (Kalfoglou and Schorlemmer 2002) created the IF-MAP method, a method for automatic ontology mapping. IF-Map generates a logic isomorphism given two ontologies. This method relies on a partial translation from the source ontologies to Horn clauses, which is then used to discover the isomorphism, if any. The result is stored for future reference. Despite providing an interesting approach to ontologies in similar domains, it does address the issue of using ontologies in different domains. That is, if no such isomorphism exist between ontologies, the IF-MAP method does not allow us to use both ontologies.

Ontology mapping and alignment has been tackled by Noy and Musen through the creation of several tools that work as plug-ins for the open-source Protégé-2000 ontology editor (Grosso, Eriksson et al. 1999). The first tool was SMART (Noy and Musen 1999), followed by PROMPT (Noy and Musen 2000) and PROMPTDIFF (Noy and Musen 2002) . The tools use linguistic similarity metrics for matching concepts. The authors claim that PROMPT not only uses linguistic similarity but also the similarities of the surrounding structures of the concepts to be merged. A set of heuristics is then applied to the performed the merging procedure. The PROMPT tool, as well as Chimaera (McGuinness, Fikes et al. 2000), provide semi-automatic guidance for the knowledge engineer. Similarly the SHOE system (Heflin, Hendler et al. 2003) provides with a set of heuristics designed to align ontologies, offering the user a set of suggestions regarding ambiguous concepts. All of these tool focus on providing guidance to the knowledge engineer, thus not supporting full automatic mapping. Furthermore, these tools create maps at the ontology level and not at partially on an on demand basis.

Another approach is to use machine learning to develop a mapping between ontologies, examples of this kind of approach are given by Lacher and Groh (Lacher and Groh 2001), with the CAIMAN system, Doan et. al. (Doan, Madhavan et al. 2004), with the GLUE system, use a set of practical similarity measures to indentify similar concepts. A Bayesian approach is used by Prasad et. al. (Prasad, Peng et al. 2002) for deciding between similarity comparisons. Even though it presents an intriguing application of several promising machine learning techniques, the GLUE system is meant to be just a component of a more encompassing tool, envisaged to have a strong human-interaction component, which would be suitable for an on demand querying system like the one we propose. The CAIMAN system considers the concepts in an ontology implicitly represented by the documents assigned to each concept. Using machine learning techniques for text classification, a concept in a personal ontology is mapped to a concept in community ontology. This is an interesting perspective but it assumes the existence of documents assigned to each concept in the ontologies, which is not the case in most available ontologies, thus making it unsuitable for a general approach.

OntoMorph (Chalupsky 2000) presents a method for translation of symbolic knowledge, integrated within the PowerLoom knowledge representation system (MacGregor, Chalupsky et al. 1997). Using syntactic rewriting through pattern matching, the author claims that the potential of this translation system is adequate to handle complicated syntactic transformations. Semantic rewriting is applied to conflate large classes of concepts. The OntoMorph system focuses on providing a rule based language that can describe complex concepts but, unlike the proposed approach, it does not focus on the retrieval and integration of the knowledge contained in heterogeneous sources.

DRAGO (Serafini and Tamilin 2005) uses the peer-to-peer paradigm with Distributed Description Logics to supply distributed reasoning services in multiple ontologies. Within what the authors call the contextual reasoning paradigm, the authors propose a distributed tableau algorithm to avoid the drawbacks of scalability and proprietary information and is able to provide with a distributed verifiability capability. DRAGO as system works by enabling a peer-to-peer network of ontologies, but it does not provide a central query point for those ontologies. Thus, each ontology can use the other ontologies, but an external user might find it difficult to access the combined knowledge in the available ontologies due to a lack of a central access point.

Piazza (Halevy, Ives et al. 2003) proposes a language based in XQuery (Boag, Chamberlin et al. 2002) that is used to describe semantic queries and that can be used with RDF style sources, although primarily developed for XML. OBSERVER (Mena, Illarramendi et al. 2000) uses interontology relationships such as synonyms, hyponyms and hypernyms to rewrite user queries to obtain translations across ontologies. A more extensive survey on the subject can be found in (Kalfoglou and Schorlemmer 2003) and in (Noy 2004). By and large, the methods for querying multiple ontologies so far hinge on the user understanding the structures that define the ontologies to be queried. Unlike the FOS method, the referred query languages don't allow the user to simply define their information need, but rather requires a knowledge of the structure of the target ontology, that is, the query must specify some portion of the concepts as well as the specific structure of the ontology.

3 Automatic Ontology Creation

Although the main focus of this thesis is the problems that arise from querying and integrating the knowledge contained in different ontologies, recent developments in semantic analysis have produced a new type of ontology that is poised to have a significant impact on the number of available ontologies. The availability of mechanisms for the automatic creation of ontological resources from unstructured text will most likely lead to a large increase in the number of available ontologies. Given the unsupervised nature of the most promising approaches (Yates, Cafarella et al. 2007) the type of ontologies and the level of noise in the information contained in those ontologies is expected to present challenges to the integration of that knowledge with current ontologies. In order to explore these types of ontologies and to allow the user to understand the issues central to this theme, this chapter focuses on the automatic creation of ontologies and the introduction of an algorithm for the extension and creation ontologies from a semi-structured source.

3.1 Automatic Creation of Ontologies from Semi Structured Resources

Most of the currently available ontologies were hand created by experts, but a new field is emerging that deals with the automatic creation of ontologies. Ideally we would like to take unstructured text and mine the knowledge contained in the text to create an ontology that describes it, similarly to the process that human experts employ in constructing ontologies. Several approaches have demonstrated some success in this area, such as in (Snow, Jurafsky et al. 2005), but general text mining for ontology building is still too noisy for use in a full automatic process. Fortunately we can address the issue by focusing on an intermediate step of sorts, the use of semi-structured text.

Large sources of encyclopedic knowledge are becoming readily available in wiki like form. Resources such as Wikipedia (Denoyer and Gallinari 2006), the largest collaboratively edited source of encyclopedic knowledge (Krotzsch, Vrandecic et al.

2005; Völkel, Krötzsch et al. 2006), Scholarpedia (Izhikevich 2007), Citizendium (Sanger 2007) and the recently launched, incipient Google Knols Project are examples of semi-structured encyclopedic knowledge bases that provide a natural way to collect human knowledge (Lih 2003), with the advantage of naturally solving, to a large degree, the problem of consensus.

These resources represent an intermediate step between unstructured text and structured knowledge and are seen as potential viable sources of knowledge for automatic construction of medical ontologies.

In this chapter we propose a general framework to mine structured knowledge from Wiki like resources and apply it to the creation of a medical ontology. The chapter proceeds as follows: we first discuss related work, and then describe the general framework for building a medical ontology from Wikipedia, presented as a test case. We demonstrate our Directional Feedback Edge Labeling algorithm on a task of labeling the relations in Okinet, a Wikipedia based medical ontology. We conclude with a description of the Okinet browser as well as some interesting and promising ideas for future work.

3.2 Related Work

Maedche(Maedche and Staab 2002; Maedche 2002) and Navigli et al.(Navigli, Velardi et al. 2003) explored semi-automatic methods for concept and relation extraction, focusing on building ontologies from broad domain documents. Blake and Pratt(Blake and Pratt 2002) worked on extracting relationships between concepts from medical texts. Khoo et al.(Khoo, Chan et al. 2002) matched graphical patterns in syntactic parse trees in order to look for causal relations.

Several pieces of previous work focused on the link structure of Wikipedia to derive structure. Kozlova (Kozlova 2005) mined the link structure in Wikipedia for document classification. Milne et al.(Milne, Medelyan et al. 2006) used the basic link structure to construct domain specific thesauri and applied it to the agriculture domain. Bhole et al.(Bhole, Fortuna et al. 2007) used document classification techniques to determine

appropriate documents in Wikipedia that were later mined for social information (people, places, organizations and events).

Powerset (Converse, Kaplan et al.), a semantic search engine recently acquired by Microsoft, uses semantic analysis on Wikipedia to provide a richer and more accurate search. Using Freebase as its knowledge base and shallow semantic parsing, including semantic role labeling, to parse Wikipedia pages, it provides the user with the ability to search based on specific roles, presenting results enriched with freebase information. Powerset's goal is to extend its techniques to general unstructured web documents, and some of its technology seems to be used in Bing, Microsoft's new search engine.

Despite promising efforts in using Wikipedia to generate semantic structures, the work so far has lacked the ability to use structural predictions to minimize the amount of data needed for labeling, on which we focus on our work.

3.3 *The Wikipedia Structure*

Wikipedia general structure consists of an *article name*, which is unique within the particular wiki structure and thus suitable for a *concept name*, and *links* connecting articles, which are suggestive of semantic relations between them. Each article is typically divided in sections and sometimes contains tables that synthesize information pertinent to the article.

Within the different types of inter-article links, we often find *redirects* (articles that consist solely of a link to another article) and when we find this type of link we can interpret the two concepts described by those articles as synonyms. Each article is normally inserted into one or more categories, thus creating a set of hierarchical relations.

Even though each link seems to carry semantic information between two concepts, only a small percentage is typically used in mining Wikipedia, namely the *redirects* and *categories*. The main challenge of this work is to assign the correct semantic label to the extracted links deemed of interest, when the link is not a redirect.

3.4 General Methodology

We propose that we should take an inclusive approach rather than a selective approach to create a medical ontology, where we start by including all the article names as concepts and all the existing links as potential relations. We subsequently rely on extracted features to assign labels, finally discarding links without labels.

The goal is to first create a directed unlabeled graph that mimics the link structure, use the extracted features to generate a small amount of labeled data and run a Directional Feedback Edge Labeling Algorithm to extend the labels to the rest of the links, discarding the links with confidence below a preset threshold.

3.5 Feature Extraction

For every link extracted we store a set of features that are associated with that link. The set of features consist of the following:

- Document Title
 - The title of the document where the link was found. This corresponds to the source concept.
- Section Header Path
 - The path composed of the sections up to the section where the link was found. E.g. Diagnosis → Symptoms.
- Context
 - The context surrounding the link. This consists of up to 3 words before and after the link.
- Link Type
 - The type of link. This can be *redirect*, *anchor*, *category* or *regular*.
 - A *redirect* link redirects the user from one page to another.
 - An *anchor* link is created when the text of the link is different from the name of the page it links to.
 - A *category* link connects an article with it's category
 - A *regular* link is any other type of link. The majority of links are of the regular type.

- Part of List
 - Binary feature that is positive if the link was found within a list, such as – *Fatigue, Headache, Nausea and Vomiting.*

In Table 1 we show an example of the information that the extraction of one link generates.

Sample Feature Extraction	
Concept	fever
Document Title	Influenza
Header Path	Symtpoms and diagnosis > Symptoms
Context	Extreme coldness and fever
Link Type	regular
Part of List	yes

Table 1. Sample Feature Extraction

Even though we extracted five features, for the purposes of this work, we used only three features. We expect to use context and header path in future work for the purpose of increasing performance.

3.6 *Generating Labeled Data*

Once we process the entire Wikipedia, we have a directed unlabeled graph where each edge represents a relation between two concepts. For each edge we also have a set of associated features.

After we decide the set of labels we are interested in, we use a combination of heuristics to bootstrap the labeling process. Besides using the redirect anchor and category links to label synonyms and hypernyms, we rely on the following two strategies.

3.6.1 List Based Labeling

Uses articles that list concepts and assigns labels to the instances of those lists that are under corresponding sections. E.g. if we find fever under the section symptoms in article flu and fever is also in the list of medical symptoms article, then we assign symptom as label for the link between flu and fever.

3.6.2 Context Based Labeling

Assigns the section (title) as label if the context shows that the link is displayed within a list. E.g. If we find *fever, headache and nausea* under the section *symptoms* under article *flu*, we assign *symptom* as a label for the link between *flu* and *fever*.

After the bootstrapping process, we have a directed graph with a partially labeled relation set. In the next section we introduce the Directional Feedback Edge Labeling Algorithm which starts with a small such set of labeled links and uses graph probability propagation to label the remaining links/relations in the ontology.

As an example, let's consider the following portions of Wikipedia pages.

Page 1 - Influenza

Symptoms and diagnosis [edit]

Symptoms of influenza can start quite suddenly one to two days after infection. Usually the first symptoms are chills or a chilly sensation, but fever is also common early in the infection, with body temperatures ranging from 38-39 °C (approximately 100-103 °F).^[52] Many people are so ill that they are confined to bed for several days, with aches and pains throughout their bodies, which are worse in their backs and legs.^[1] Symptoms of influenza may include:

- Body aches, especially joints and throat
- Extreme coldness and fever
- Fatigue
- Headache
- Irritated watering eyes
- Reddened eyes, skin (especially face), mouth, throat and nose
- Abdominal pain (in children with influenza B)^[53]

It can be difficult to distinguish between the common cold and influenza in the early stages of these infections,^[2] but a flu can be identified by a high fever with a sudden onset and extreme fatigue. Diarrhea is not normally a symptom of influenza in adults,^[51] although it has been seen in some human cases of the H5N1 "bird flu"^[54] and can be a symptom in children.^[55] Research on signs and symptoms of influenza found that the best findings for excluding the diagnosis of influenza (having ratios less than 0.5 between the likelihood of having flu if one does not have the symptom to the likelihood if one does have the symptom) were:^[51]

Highest sensitive individual findings for diagnosing influenza^[51]

Finding:	sensitivity	specificity
Fever	68-86%	25-73%
Cough	84-98%	7-29%
Nasal congestion	68-91%	19-41%

Notes to table:

- The ranges given represent different studies that were reviewed.
- Sensitivity is the proportion of people having influenza who exhibit the symptom.
- Specificity is the proportion of people not having influenza who do not exhibit the symptom.
- All three findings, especially fever, were less sensitive in patients over 60 years of age.

Since anti-viral drugs are effective in treating influenza if given early (see treatment section, below), it can be important to identify cases early. Of the symptoms listed above, the combinations of findings below can improve diagnostic accuracy.^[56] Unfortunately, even combinations of findings are imperfect. However, Bayes Theorem can combine pretest probability

Page 2 – Leukemia

- Português
- Русский
- Shqip
- Simple English
- Slovenčina
- Српски / Srpski
- Sрpskohrvatski / Српскохрватски
- Basa Sunda
- Suomi
- Svenska
- ไทย
- Tiếng Việt
- Türkçe
- Угърська
- 中文

Symptoms [edit]

Damage to the bone marrow, by way of displacing the normal bone marrow cells with higher numbers of immature white blood cells, results in a lack of blood platelets, which are important in the blood clotting process. This means people with leukemia may easily become bruised, bleed excessively, or develop pinprick bleeds (petechiae).

White blood cells, which are involved in fighting pathogens, may be suppressed or dysfunctional. This could cause the patient's immune system to be unable to fight off a simple infection or to start attacking other body cells. Because leukemia prevents the immune system from working normally, some patients experience frequent infection, ranging from infected tonsils, sores in the mouth, or diarrhea to life-threatening pneumonia or opportunistic infections.

Finally, the red blood cell deficiency leads to anemia, which may cause dyspnea and pallor.

Some patients experience other symptoms. These symptoms might include feeling sick, such as having fevers, chills, night sweats and other flu-like symptoms, or feeling fatigued. Some patients experience nausea or a feeling of fullness due to an enlarged liver and spleen; this can result in unintentional weight loss. If the leukemic cells invade the central nervous system, then neurological symptoms (notably headaches) can occur.

All symptoms associated with leukemia can be attributed to other diseases. Consequently, leukemia is always diagnosed through medical tests.

The word *leukemia*, which means 'white blood', is derived from the disease's namesake high white blood cell counts that most leukemia patients have before treatment. The high number of white blood cells are apparent when a blood sample is viewed under a microscope. Frequently, these extra white blood cells are immature or dysfunctional. The excessive number of cells can also interfere with the level of other cells, causing a harmful imbalance in the blood count.

Some leukemia patients do not have high white blood cell counts visible during a regular blood count. This less-common condition is called *aleukemia*. The bone marrow still contains cancerous white blood cells which disrupt the normal production of blood cells. However, the leukemic cells are staying in the marrow instead of entering the bloodstream, where they would be visible in a blood test. For an aleukemic patient, the white blood cell counts in the bloodstream can be normal or low. Aleukemia can occur in any of the four major types of leukemia, and is particularly common in hairy cell leukemia.

Causes and risk factors [edit]

There is no single known cause for all of the different types of leukemia. The different leukemias likely have different causes. Known causes include natural and artificial ionizing radiation, viruses such as Human T-lymphotropic virus, and some chemicals, notably benzene and alkylating chemotherapy agents for previous malignancies.^{[7][8][9]} Use of tobacco is associated with a small increase in the risk of developing acute myeloid leukemia in adults.^[7] A few cases of maternal-fetal transmission have been reported.^[7]

Leukemia, like other cancers, results from somatic mutations in the DNA which activate oncogenes or deactivate tumor suppressor genes, and disrupt the regulation of cell death, differentiation or division. These mutations may occur spontaneously as a result of exposure to endogenous substances and are likely to be influenced by genetic factors.

From these pages, by applying the heuristics described, we would extract the following relationships, not shown in full here in benefit of clarity. In this case we focus on the *symptom_of* relationship

From Page 1

- Headache → symptom_of → influenza
- Fever → symptom_of → influenza
- Fatigue → symptom_of → influenza
- Abdominal Pain → symptom_of → influenza

From page 2

- Infection → symptom_of → leukemia
- Fatigue → symptom_of → leukemia
- Diarrhea → symptom_of → leukemia
- Penumonia → symptom_of → leukemia
- Headache → symptom_of → leukemia
- Fever → symptom_of → leukemia

From these links, we could construct the following symptom_of ontology.

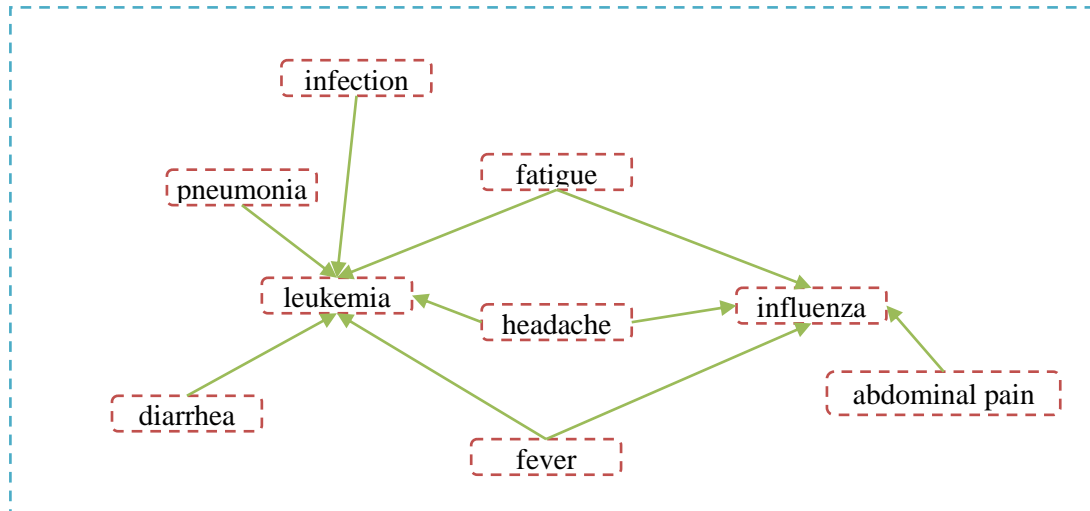


Figure 3 - Ontology Extraction Example

As we can see, even with this tiny ontology, we are already able to answer the query “What diseases have fatigue, headache and fever as common symptoms?” In this case we can produce a set of labeled relations, which will be expanded using the Directional Edge feedback algorithm.

3.7 Directional Feedback Edge Labeling Algorithm

The Directional Feedback Edge Labeling Algorithm relies on neighboring edge trends and directionality to update the confidence of possible labels that can be assigned to an unlabeled relation. The steps of this algorithm are described in Algorithm 1. It is important to note this algorithm assumes that the each link can only be labeled by one of a finite set of known labels.

Each unlabeled edge starts with equal probability of label assignment. At each iteration, in STEP 1, for each node we update the probabilities of the labels of the outgoing edges by smoothing them with the overall probability distribution of labels over the outgoing edges of that node (essentially multiplying the two probability distributions). This assures we take into account both our current belief about that edge and the overall information contained in the edges going out of that node. To give an intuition why both types of information are important, consider the example in Figure 1. The dashed and the dotted

edges represent edges which were labeled during the bootstrapping phase. The dashed edges represent label *SymptomOf* and the dotted edges represent label *Treats*. The solid edges are unlabeled and therefore it is natural to assume that, in the absence of other information, each label is equally likely. However, based on the already labeled outgoing edges at C_1 , the unlabeled edge (C_1, C_{10}) has a $2/3$ probability to have label *SymptomOf* and $1/3$ probability to have label *Treats*. Therefore, our initial belief of the edge (C_1, C_{10}) needs to be updated by incorporating this new information.

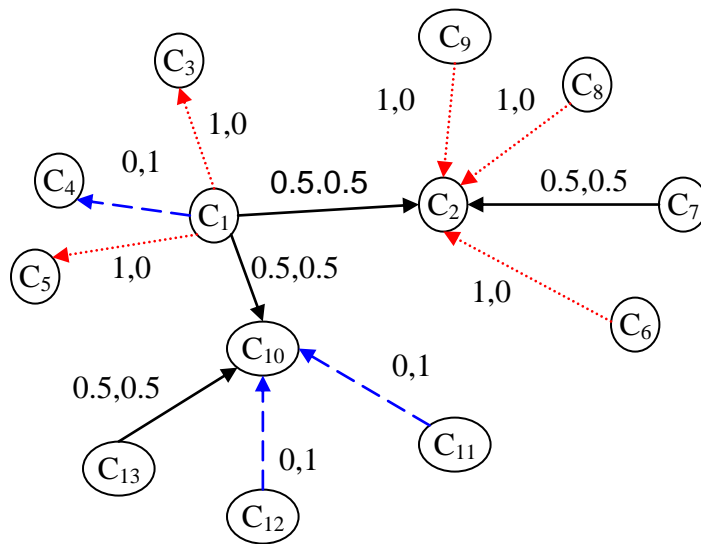


Figure 4 - Example of Directional Labeling.

For each node C , perform Steps 1 and 2, then repeat until convergence.

STEP 1. Let p_{ik} be the probability of the i^{th} outgoing edge (out of n possible) from node C to have the k^{th} label (out of m possible labels). Update the outgoing edge probabilities:

$$P_{ik} \leftarrow \frac{P_{ik} \times \sum_{j=1}^n P_{jk}}{\sum_{l=1}^m (P_{il} \times \sum_{j=1}^n P_{jl})}$$

STEP 2. Update the incoming edge probabilities similar to the previous step.

STEP 3. Once convergence is reached via the above two steps, assign the maximum probability label to an edge as long as this probability is higher than a predefined threshold.

Algorithm 1. Directional Feedback Edge Labeling Algorithm.

In STEP 2, we then perform the same procedure for each node, but based on incoming edges. Because an edge is an incoming edge for a node and an outgoing edge for another, the label probability distribution for that edge is influenced by the label distributions in both its endpoints. Therefore, after a number of iterations, the label probabilities can be influenced by other label probabilities at any distance in the graph.

Back to the example in Figure 1, the edge (C_1, C_{10}) has a $2/3$ probability to be labeled *SymptomOf* if we look only at the outgoing edges from C_1 whereas it has a probability of 1 to be labeled *Treats* if we look only at the incoming edges to C_{10} . This justifies the need to perform the same operation for both incoming and outgoing edges. The need to perform both steps iteratively is twofold: to assure convergence and to allow knowledge to propagate across the network. After convergence, we select only the edges with labels above a predefined threshold and discard the rest as unreliably labeled.

3.8 UMLS and Okinet

UMLS is perhaps the most important medical ontology currently available. It uses a semantic network to combine the knowledge contained in the set of available dictionaries and allows for easy access to a set of standard ontological relations. The work of mapping the vocabularies demands a large human effort and is very time consuming. Due to the structure of UMLS, certain semantic relations exist only at the semantic network level. This means that in UMLS we are not able to determine symptoms of particular diseases, but rather between classes of concepts. For example, we are not able to find out what are the *symptoms_of flu*, but rather what categories of concepts could represent symptoms of *flu*, which is a problem.

Okinet is a Wikipedia-based (Pedro, Niculescu et al. 2008) ontology and is currently being developed at Siemens Medical Research. Automatically extracting an ontology from Wikipedia presents unique challenges since the underlying data is very large and the source is updated frequently. Compared to previous automatic ontology extraction work, we use a combination of feature based and graph based approaches to ontology building. Okinet currently contains more than 4 million concepts and encodes a wide range of semantic relations both traditional semantic relations such as synonyms, hypernyms and as instance based semantic relations such as medical specific relations such as *symptom_of*, *causes*, *medication*, etc. Okinet is high coverage (open domain ontology), high depth, low cost (existing knowledge), high noise (automatic processes) and high connectivity ontology.

We built Okinet as a complement to UMLS, allowing the rapid and automatic creation of relations at the instance level, which enables the use of inference processes using both ontologies.

3.9 Experimental Setting

In order to test our approach, we used Wikipedia as a test case, even though the methodology could be applied to any other wiki like resource. Our goal is to create an ontology of causes, treatments, symptoms, diagnoses and side effects.

We started by selecting all the concepts contained in the *list of diseases* article, which contains 4000+ diseases and syndromes. We then expanded our article set to include all the articles that linked or were linked to by any of the articles contained in the current set.

Next we performed the feature extraction process followed by the bootstrapping procedure. The results were manually checked to create a gold standard set. This resulted in an ontology with 4308 concepts and 7465 relations divided as depicted in Figure 2.

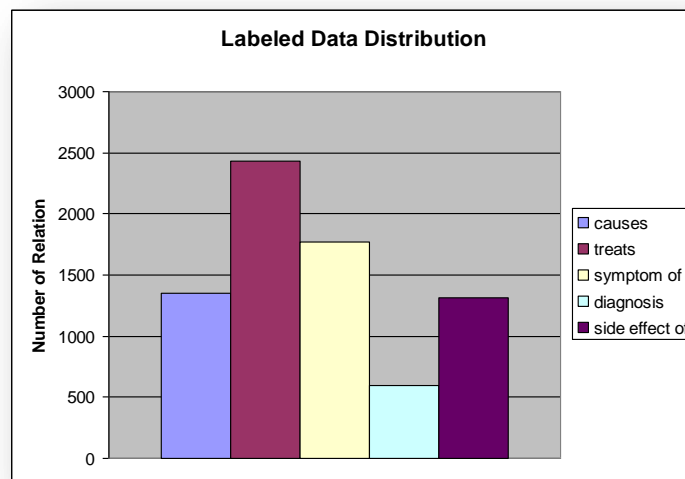


Figure 2. Distribution of relation labels.

3.10 Results

We experimented using a variable percentage of the labeled data, between 10% and 90%, as a training seed for the Directional Edge Feedback Labeling algorithm while considering the remaining edges unlabeled. The results of the labeling algorithm were then compared with the original labels. In our experiments, we varied both the percentage

of the labeled training data (seed size) as well as the threshold above which we assign a label. We evaluated the results using precision and recall:

Precision The percentage of label assignments that were correctly assigned to the proper class.

Recall The percentage of possible labels that were correctly assigned.

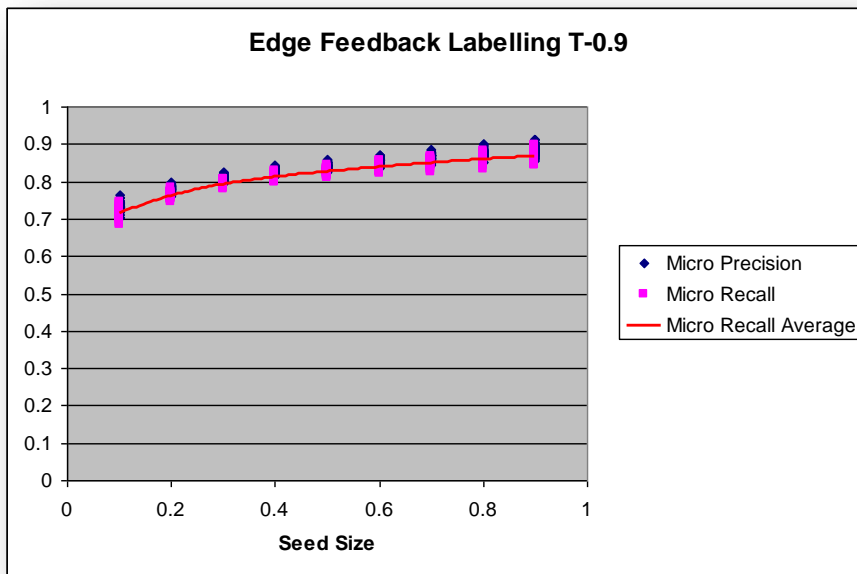


Figure 3. Algorithm performance with threshold 0.9 and variation on the seed size

In Figure 3 we can see the results of varying the size of labeled seed set at a threshold for label assignment at 0.9, which means that we only assign a label with high confidence. Even though we are only showing micro precision and micro recall, the results for macro precision and recall were very similar and were thus not presented for simplicity purposes. Each point in the average line represents a run with a labeled seed size of the indicated value. For each seed size we ran 100 runs. The precision and recall average vary between 70% and 90% while seeds vary from 10% to 90% of the total labeled set.

Even though the results are very promising, we explored ways to boost the results at small seed sizes. Due to the propagation nature of this algorithm, by stopping after a few

iterations, we are in fact preventing long-range labeled edge influences and therefore we can restrict the process of labeling an edge to local neighborhood in the graph. Figure 4 shows the results of stopping the labeling algorithm after two iterations. Given the number of iterations, only edges with a fast convergence rate will update the probabilities distribution enough to get assigned a label. This means that the higher the threshold, the more accuracy we get, even though the recall is sharply reduced. This variation is particularly useful in situations where the precision is more important than recall. Using this technique we would be able to extend the labeled data set with highly accurate labels.

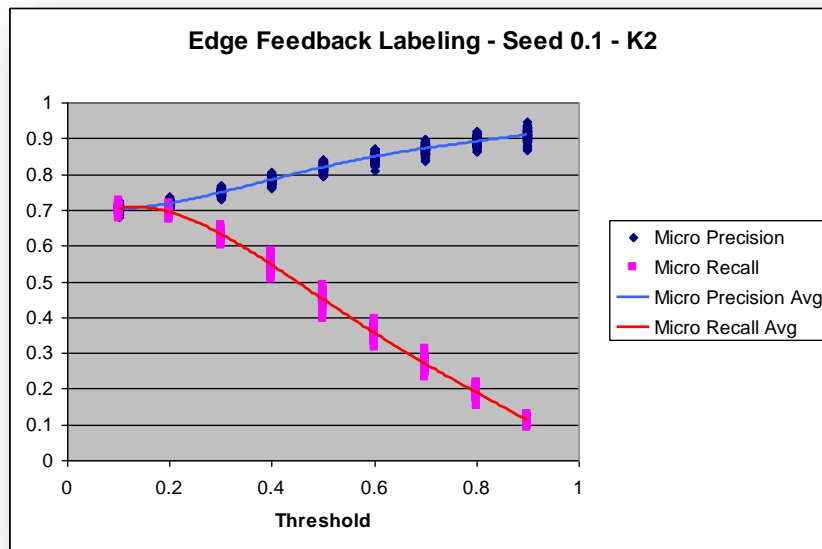


Figure 4. Precision and recall with 10% seed size, algorithm stopped after two iterations and varying the assignment threshold.

Finally we looked at our algorithm as way to reduce uncertainty. Figure 5 shows the results of taking the two highest confidence labels for each edge and considering as correct if either of the assigned labels is correct.

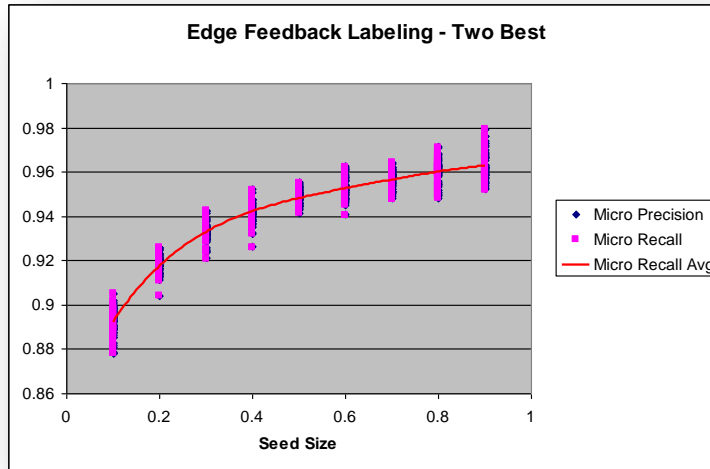


Figure 5. Precision and Recall when considering looking at the two labels with the highest probability

4 Ontology Selection

In this chapter we will focus on the sub problem of ontology selection. Ontology selection deals with selecting the appropriate resources to query given a specific query. As the number of ontologies grows so does number of ontologies that contain information that might be considered pertinent for a given query. The first question one must address is how to decide if an ontology does indeed contain pertinent information given a query. This problem is not the main focus of this thesis and as such the approaches taken are not exhaustive. Even though we will discuss several possible approaches, a lot of the work will be left as future work. Our contribution here is twofold. On one hand we propose a characterization methodology for ontologies. On the other hand we discuss the possible scenarios and the variables that are relevant to the problem and suggest different approaches to the problem of ontology modeling, while taking a pragmatic approach for practical considerations.

The chapter is organized as follows. We will start by proposing a general method for Ontology Characterization and describing current ontological representation languages. We then propose an approach to ontology modeling and selection and discuss the practical implications of such approach.

4.1 Characterization of Ontological Resources

The challenge of retrieving and merging knowledge from multiple ontologies is amplified by the fact that to date there are no simple methodologies to describe and classify available ontologies. Even in vertical domains, such as the medical domain, there is a considerable diversity in the resource space, which includes highly focused resources (e.g. radiology-specific ontology), broad collections of resources (e.g. UMLS), and institution-specific ontologies. These differ drastically in terms of coverage, relevance, and scope.

We propose a more formal characterization of ontological resources which describes resources according to the following dimensions

1. **Scope** (or coverage) is the percentage of broad semantic topics covered by the ontology. The broader the coverage the more diverse is the ontology. The scope of an ontology can be estimated using different methods (Brank, Grobelnik et al. 2005), for example accounting for loosely connected components, that is, counting the number of clusters which are semi isolated from the rest.
2. **Depth** is the amount and complexity of knowledge encoded in the ontology on a particular subject. The depth of an ontology can be estimated, for example, using:
 - a. The average path length in the ontology. The higher the average path the more specialized is the ontology
 - b. The average number of concepts in each semantic topic, as defined in Scope.
3. **Cost** of building the ontology can be determined by a function $f(T_c, C_c)$, where T_c is the average time spent per ontology node (concept) and C_c is the (financial) effort in constructing it. The range varies from fully automatic ontologies, which require almost no human effort, to specialized, fully manually generated ontologies - which take considerable time to be built and are expensive in terms of human expertise.
4. **Noise** is the percentage of concepts in the ontology that carry no semantic meaning. Given a set of concepts, a human should be able to determine which concepts carry semantic meaning. This issue pertains especially to automatically created ontologies, which tend to introduce semantically irrelevant concepts because of the extraction process.
5. **Connectivity** describes how connected the average concept is to other nodes, as well as the variety of relation types that the ontology contains. This can be given by the average number of relation types connecting each concept to the rest of the ontology

These measures were designed to be of a general nature. They can be used to characterize any ontology and provide meaningful statistics in the selection/merging process. The main use within the Federated Ontology Search Approach for the Ontology Categorization Measures is the creation of default confidence measures. When we find an unseen ontology, we use these measures to get a default confidence value for the ontology. The intuition behind this approach is that there is a correlation between the measures, sometimes an inverse correlation. E.g. Ontologies that are generated automatically typically present a high level of noise and low cost level. The more expensive and depth an ontology has, the higher the confidence in the knowledge it contain, since there is usually less ambiguity and noise.

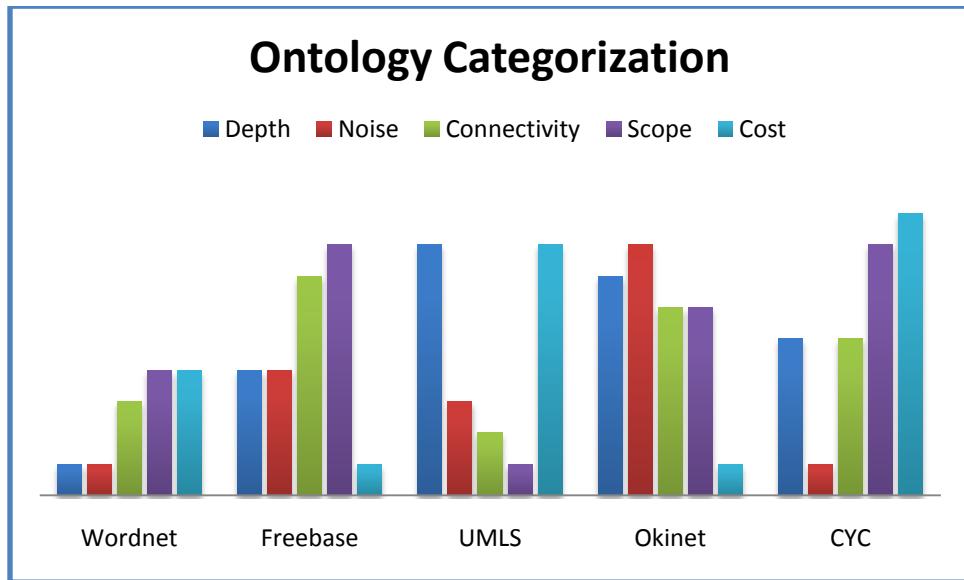


Figure 5 - Sample Ontology Categorization

Figure 5 shows the description of several ontologies according to proposed categorization scheme. Five ontologies were chosen to exemplify the range of the different characteristics: Wordnet, Freebase, UMLS, Okinet and CYC. These ontologies are described below. In Table 5 we place them in the newly defined description space.

	Depth	Noise	Connectivity	Scope	Cost
Wordnet	Low	Low	Medium	Medium-High	Medium-High

Freebase	Medium-High	Medium-High	High	High	Low
UMLS	High	Medium	Medium-Low	Low	High
Okinet	High	High	High	High	Low
CYC	Medium-High	Low	Medium-High	High	High

Table 5 - Ontology Categorization

Although the exact score of each ontology is subjective, this scale is meant to describe ontologies in a broad sense. Generally speaking it should be able to compare any two ontologies given these dimensions. We consider that there are possibly many characterization schemes for ontologies, although they are contradictory with the one proposed here. We now describe what we consider to be clear examples of different ontology types according to the characterization given above.

Wordnet (Miller 1995) is a semantic lexicon for the English language and one of the most widely used general ontologies. It groups words into sets of synonyms, provides succinct definitions, and records various semantic relations among synonym sets. It currently holds 155327 unique concepts and relations. It constitutes the canonical example of an ontology with high coverage, low depth, medium cost, low noise and medium connectivity.

Freebase (Bollacker, Evans et al. 2008) is a highly structured collaborative online database and website developed by Metaweb. Freebase contains data harvested from sources such as Wikipedia and MusicBrainz, as well as individually contributed data from its users. It represents a new paradigm on ontology creation. It contains a large number of relation types. Freebase represents a large collaborative ontology example and has high coverage (a large number of potential topics), Low cost (most of the knowledge is generated automatically, medium noise (the knowledge entered by users tends to lead to ambiguity, medium depth and high connectivity (there is a large number of relation types).

UMLS (Unified Medical Language System) (Bodenreider 2004) is both an ontology and a set of ontologies. It aims to bring together a set of medical dictionaries (currently holding roughly 100) and contains information about medical concepts, semantic types, and the relations between concepts and semantic types. It is manually maintained and it is the result of a 15 year long project that involved on the order of thousand of man hours. It currently contains over 1 million concepts. UMLS represents the largest effort of manual ontology mapping to date. Due to its specific nature and medical domain expertise, UMLS has low coverage (focuses on the medical domain), high depth (it is very specialized), high cost (highly specialized knowledge providers), low noise (human-inserted concepts have semantic meaning) and low connectivity (only very relevant relations).

In our previous work we created **Okinet** (Pedro, Niculescu et al. 2008), a Wikipedia-based ontology currently being developed at Siemens Medical Research. Automatically extracting an ontology from Wikipedia presents unique challenges since the underlying data is very large and the source is updated frequently. Compared to previous automatic ontology extraction work, we use a combination of feature based and graph based approaches to ontology building. Okinet currently contains more than 4 million concepts and encodes a wide range of semantic relations both traditional semantic relations such as synonyms, hypernyms and as instance based semantic relations such as medical specific relations such as symptom_of, causes, medication, etc. Okinet is high coverage (open domain ontology), high depth, low cost (existing knowledge), high noise (automatic processes) and high connectivity ontology.

In development since 1984, the **Cyc Knowledge Base (KB)** (Lenat 1995) is a general-purpose repository of common sense and specialized knowledge consisting of over 328,000 concepts and over 3,500,000 explicitly declared assertions. Knowledge in Cyc is represented in CycL, a higher-order logical language based on predicate calculus. Every CycL assertion occurs in a context, or *microtheory*, allowing for the representation of competing theories, hypotheses, and fictional claims. Cyc's inference engine combines

general theorem proving (rule chaining) with specialized reasoning modules to handle commonly encountered inference tasks, such as transitivity.

4.2 Ontological Representation Languages

A large set of representation languages is available for ontology representation. We do not intend to exhaustively describe all the available languages, but rather to describe and compare some of the most widely used and those with particular significant relevance to the work described in this thesis. For a more in depth look at the particularities of ontological languages, the reader is directed to (Gómez-Pérez and Corcho 2002) and (Staab 2004).

4.2.1 Knowledge Interchange Format (KIF)

KIF (Genesereth and Fikes 1992) is a computer-oriented language for the interchange of knowledge among disparate programs. It has declarative semantics (i.e. the meaning of expressions in the representation can be understood without appeal to an interpreter for manipulating those expressions); it is logically comprehensive (i.e. it provides for the expression of arbitrary sentences in the first-order predicate calculus); it provides for the representation of knowledge about the representation of knowledge; it provides for the representation of nonmonotonic reasoning rules; and it provides for the definition of objects, functions, and relations. It was a direct result of the DARPA Knowledge sharing effort and achieved some success as a language for knowledge transfer.

4.2.2 Resource Description Framework (RDF)

RDF is a metadata data model based on the idea of making statements about web resources in the form of subject predicate-object expressions, called triples in the RDF terminology. The subject indicates the resource; the predicate denotes a trait or aspect of the resource and basically expresses a relationship between the subject and object.

RDF is a major component of the W3C's Semantic web effort. Several additional languages can be built on top of Rdf, such as RDFS and OWL. Bellow is an example of RDF. Table 6 shows the content in a table form and Figure 6 shows the RDF representation of the same content.

Table 6 - RDF Example in table Form

Title	Artist	Country	Company	Price	Year
Empire Burlesque	Bob Dylan	USA	Columbia	10.90	1985
Hide your heart	Bonnie Tyler	UK	CBS Records	9.90	1988

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">
<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
</rdf:Description>
<rdf:Description
rdf:about="http://www.recshop.fake/cd/Hide your heart">
  <cd:artist>Bonnie Tyler</cd:artist>
  <cd:country>UK</cd:country>
  <cd:company>CBS Records</cd:company>
  <cd:price>9.90</cd:price>
  <cd:year>1988</cd:year>
</rdf:Description>
</rdf:RDF>
```

Figure 6 - RDF Example in XML notation

4.2.3 Web Ontology Language (OWL)

OWL (McGuinness and van Harmelen 2004) is family of knowledge representation languages for authoring ontologies. Owl is a W3C's endorsement and is based in two semantic, namely OWL DL and OWL Lite, which in turn are based in Description Logics (Baader and Nutt 2003). OWL is currently considered one of the main technologies underpinning the semantic web effort and has demonstrated academic value as well as commercial value.

The main intuition behind OWL is the interpretation of OWL as a set of individuals and a set of property assertions. Basically OWL allows the introduction on constraints on top of RDF Graphs and the introduction of relations between classes of individuals.

- OWL Lite supports primarily a classification hierarchy and simple constraints. It supports cardinality constraints, but only with values of 0 or 1.
- OWL DL is the most expressive version of the OWL family that is still able to guarantee computational completeness and decidability.

Below is an example of OWL. In this example we are setting a constraint on the property name of the Airport class. We are stating that all values for the property name must be of type string.

```
<rdfs:Class rdf:ID="Airport">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#name"/>
      <owl:allValuesFrom rdf:resource=
        "http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</rdfs>
```

Figure 7 - OWL Example

4.2.4 CycL

CycL (Lenat and Guha 1991) is the ontology language used by Doug Lenat's artificial intelligence project. CycL is a declarative language based on first-order logic. It contains extensions for modal operators and higher order quantification. CycL is used to represent knowledge contained in CYC (Lenat 1995). CycL has the following main concepts:

- The statement of general rules that support inference about concepts.
- The truth or falsity of a statement is context subjective. CYC uses the concept of microtheories to define contexts.

The main predicates in CYC are the *#\$isa* and the *#\$genls*. *#\$isa* describes the instance relationship between some collection and an item. *#\$genls* describes the relationship of subcollection between two collections.

Below is an example of the transitivity rule in CycL.

```
(#$implies
  ($and
    ($isa ?OBJ ?SUBSET)
    ($genls ?SUBSET
?SUPERSET))
  ($isa ?OBJ ?SUPERSET))
```

Figure 8 - CycL Example

Figure 7 is a CycL example that states that if *A* is an instance of *B* and *B* is sub collection of *C*, then *A* is an instance of *C*.

4.3 Proactive Ontology Selection

The increasing trend in the availability of resources suggests that often we will be able to find overlapping sources for a given query. Ideally we would like to be able to select the resources to query in order to maximize the probability of success. It is important to find the right tradeoff between querying a sufficient number of ontologies, thus maximizing the amount of relevant information retrieved, and reducing the number of ontologies queried, thus minimizing querying overhead. The more ontologies we query the more likely is that we are increasing result ambiguity, merging and result complexity.

4.3.1 Cooperative Ontology Selection

A cooperative source is a source whose knowledge is fully available for querying and indexing. In many cases though, it's not realistic and maybe not even desirable to expect cooperative sources. The proprietary content in some ontologies might not be made available by its authors, or perhaps only part of the ontology might be available, with filters to control access to the information contained in such ontology. Examples of this

can be taken from CYC, which releases OpenCyc as a free smaller portion of the knowledge contained in the full CYC. Although at this moment they are separate entities, one could conceive of a controlled access paradigm. We must also consider cases where the ontology has incorporated inference engines and logic mechanisms, the use of which is advantageous and important.

For an example of the importance of resource description, we refer to the example in [cite reference here to the example], where selecting the wrong resource would lead to wrong results, Where without an adequate resource description, it will be extremely hard to differentiate between expert and non-expert sources.

In the case of cooperative sources the problem can be satisfactorily addressed by indexing the ontology using a search engine such as the Lemur Search Engine (Avrahami, Yau et al. 2006). This approach is taken by initiatives such as the Freebase approach, where the search operation, rather than query the ontology directly, queries the indices created over the ontological concepts. This presents a good good strategy, since search engines are fairly mature and primed for scaling and performance. The result is a very fast access to concepts contained in the ontology,

4.3.2 Uncooperative Ontology Selection

While it seems that there is a trend in ontological work to assume that ontological resources are cooperative resources, this is not always the case and in fact it is not desired. Cooperative sources allow for full indexation of the knowledge, but have a couple of disadvantages. Having the data locally can have higher maintenance issues, e.g. when a new version of the ontology is available, and typically we lose access to the inference engines that were created specifically for that ontology. Also, it might not be even possible to have the whole data, in the case of proprietary ontologies such as CYC. By creating a model that is able to deal with uncooperative sources, that is sources that are not entirely available for indexing, we are able to punt the maintenance issue to the ontology owner, use specific inference engines that might have been created for that

ontology and we are able to foster an distributed environment where each person is encouraged to create his or her own ontologies. By maintaining autonomy of ontologies users can profit from content creation directly, since it is possible to determine the relevance of an ontology for each query, which would allow for models of revenue sharing according to relevance, which in turn would foster the creation of more knowledge.

The main problem thus becomes the selection of the right ontologies in situations where we are dealing with both cooperative and uncooperative sources.

In order to tackle this problem we take inspiration in the work by Donmez and Carbonell (Donmez and Carbonell 2008) and apply the concept of proactive learning to Ontology Selection. Proactive learning is basically a generalization of active learning where each oracle is modeled as having different properties that are taken into account at labeling time. A similar strategy can be applied to ontology selection. The intuition behind this idea is that each ontology is considered an expert in some domain, like an oracle, that has certain properties to it, namely noise, domain, level of expertise, etc. The goal, like in Proactive learning, is to select the best set of ontologies at query time. This goal is accomplished by modeling the ontologies a priori using a set of sample queries. The queries are then clustered according to a clustering algorithm such as the k-medoid clustering algorithm (Kaufman and Rousseeuw 1990) and new queries are compared against the samples in order to choose the best set of ontologies.

4.3.2.1 Sample extraction for ontology modeling

In order to model the available ontologies regarding the potential for answering queries, we must create a set of sample queries that we can have the ontologies answer in order to estimate the correct parameters of each ontology, thus taking a similar approach to the work in federated search by Si and Callan in (Si and Callan 2005). In our case, the main problem in creating the sample query set is the fact that query operators must be taken into account when creating the queries, which makes the creation of a query sample set

that is truly representative an impossible task, since there are potentially infinite operator combinations possible due to the combinatorial nature of the query operators.

We can begin to approximate the problem by focusing on the search operator and a simple relation operator expression, thus creating a sample set that we can use as base to approximate to other queries to a large extent. The sample queries sets are produced sequentially, first creating the sample set for the search operator and then using the results to create a set of relational operators.

The main idea is to take a dictionary and use the dictionary entries to generate a set of search queries. That is, for every word x in dictionary D , we generate query q_x such that $q_x = \text{"Search}(x)\text{"}$.

Then for each q_x and for each ontology O we generate $R_O(q_x)$, that is the result of executing q_x in O . Let's say for example that the dictionary D contains the words JAG and JAGUAR. The queries generated in this case would be $q_1 = \text{Search}(JAG)$ and $q_2 = \text{Search}(JAGUAR)$ respectively. Let's say that we have O_1, O_2, O_3 and O_4 as our ontologies, with domains D such that

Table 7 - Domains of sample ontologies

Domains	
$D_{O_1} =$	CARS
$D_{O_2} =$	ANIMALS
$D_{O_3} =$	TV SERIES
$D_{O_4} =$	FRUITS

We would then take q_1 and q_2 and get the following hypothetical results. Each result consists of a concept that can be represented by the query string.

Table 8 - Sample Query Result

Result	
$R_{O_1}(q_1) =$	JAGUAR (CAR)
$R_{O_1}(q_2) =$	JAGUAR (Animal)
$R_{O_2}(q_1) =$	Null

$R_{o_2}(q_2) =$	JAGUAR (ANIMAL)
$R_{o_3}(q_1) =$	JAG (TV SERIES)
$R_{o_3}(q_2) =$	Null
$R_{o_4}(q_1) =$	Null
$R_{o_4}(q_2) =$	Null

As we can see in Table 8, not all ontologies return results. The last step is to take the results obtained in the previous step and create the relationship queries.

For every concept c_1, c_2 we create a relationship query $Rel(c_1, c_2)$ and again run it in the available ontologies.

Thus starting with a set of words $W = \{w_1, \dots, w_n\}$ we get a set of search queries $Q_s = \{q_{w_1}, \dots, q_{w_n}\}$, that returns a set of concepts $C = \{c_1, \dots, c_n\}$, which in turn generate a set of relationship queries $Q_r = \{q_{(c_1, c_1)}, \dots, q_{(c_n, c_n)}\}$. The query sets Q_s and Q_r are the sample sets that we will use to model each ontology. For every query q in Q_s or Q_r , we select as the target ontologies those that return non-empty results.

4.3.2.2 Run Time Ontology Selection

After creating the sample query sets and running them against the available ontologies, the process of selecting the appropriate ontologies for query q_t at run time is done by selecting the query from the sample that is closest to q_t and querying the ontologies that returned non-empty result sets for the sample query that we selected.

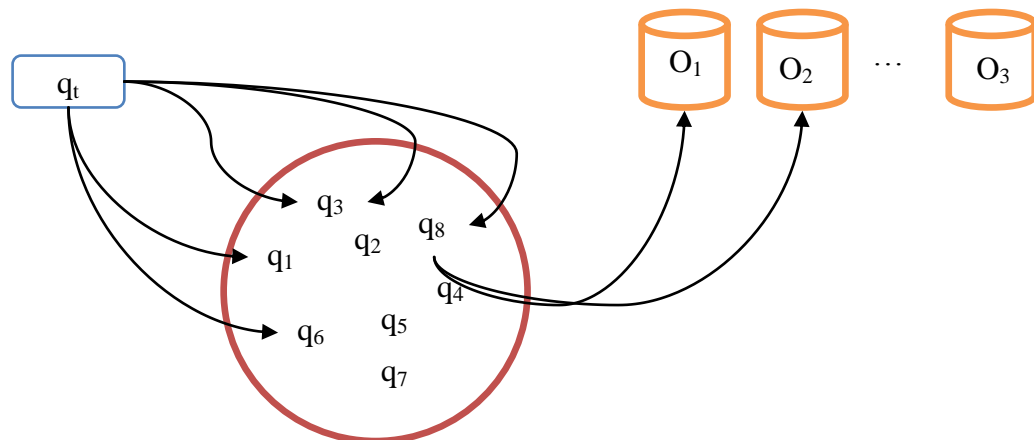


Figure 9 - Overview of query Run Time ontology Selection

There are two types of primary queries, *search queries* and *relation queries*. Search queries deal with the relationship of string and concepts. Relation queries deal with the relations between concepts. The set that is used depends of the type of query. If we are performing a *search* query then we use the search sample query set, otherwise we use the relation sample query set. In order to find the closest query to q_t , we can use any number of similarity algorithms. In the case of the *search* operator we must use string based methods, since we are comparing strings. That is, given query q_t we compare it with all sample queries q_1, \dots, q_n using a method like the Q-Gram measure (Gravano, Ipeirotis et al. 2001) or any number of other string similarity measures.

When comparing relation queries, it is important to note that we are now dealing with concepts rather than strings, so we should use semantic similarity methods as described in (Espinoza, Trillo et al.) The goal is to use a subset of the available ontologies to measure the semantic distance between the query and the cluster medoids. Medoids are representative objects of a data set or a cluster with a data set whose average dissimilarity to all the objects in the cluster is minimal. Medoids are similar in concept to means or centroids, but medoids are always members of the data set. We use the medoids since comparing to all possible queries is computationally expensive. This is achieved by measuring the semantic distance between each of the concepts in the query. This relies on the assumption that certain general ontologies such as wordnet will always be available. For example, let us say that we have two queries q and q' .

$$q = rel(car, driver)$$

$$q' = rel(automobile, driver)$$

The semantic distance D_s between q and q' can be expressed as

$$D_s(x, z) = \text{Distance in hops from } x \text{ to } z \text{ in the chosen ontology}$$

or

$$D_s(rel(x, y), rel(z, w)) = \frac{D_s(x, z) + D_s(y, w)}{2}$$

That is

$$D_s(q, q') = \frac{D_s(car, automobile) + D_s(driver, driver)}{2}$$

The distance is then calculating by traversing the designated general ontology and returning the number of hops, or null, if the two terms are not connected.

If the available ontology does not contain one or more concepts that we are trying to compare, then we back off to taking the concept and treating it individually as a query of type *Search(x)*. That is, in case of query *q* describe above, if the ontology does not have knowledge about *car*, we then query all the ontologies for *search(car)*. This will give us, in the worst case scenario, a set of ontologies that is larger than the optimal set, but it expands our coverage significantly.

5 Ontological Search

In this chapter we will focus on the sub problem of the query language used for querying a federated system of ontologies and the process of ontology selection. We will start by describing current query languages and briefly discuss the advantages and shortcomings of the described languages regarding a federated approach to ontology search. We will then describe the proposed operator based query language and finally we will address the issue of ontology selection for a given query.

5.1 *Ontology Query Languages*

A distinction must be made between representation and query languages. With the exception of CycL, which is used to both represent and query CYC content, all the representation languages described in the previous chapter are used specifically for representation purposes. Specific query languages are typically used to query the information contained in the ontologies that are represented in one of the previously described representation languages. It is not the scope of this thesis to describe each query language exhaustively, but rather to expose the reader to the most important, the most commonly used and the most relevant to the work described in this thesis. For a more in-depth look at ontology query languages the reader is directed to (Staab 2004).

5.1.1 *Simple Protocol and RDF Query Language (SPARQL)*

SPARQL (McCarthy 2005) is quickly becoming the de facto language to query RDF graphs. It is the W3C candidate recommendation query language for RDF and it is considered by the W3C as a vital component of the semantic web. SPARQL allows for a query that consists of triple patterns, conjunctions, disjunctions and optional patterns. SPARQL does not have a native inference mechanism incorporated into the language. SPARQL queries return only what is contained in the information model in the form of graph bindings. Figure 10 shows a SPARQL query that returns all capitals of country within Europe.

```
PREFIX pref: <http://example.com/exampleOntology#>
SELECT ?capital ?country
WHERE {
  ?x pref:cityname ?capital ;
     pref:isCapitalOf ?y .
  ?y pref:countryname ?country ;
     pref:isInContinent abc:Europe .
}
```

Figure 10 - SPARQL Example

Variables are indicated by the “?” or “\$” prefix. In the query shown in Figure 10, bindings for *?capital* and *?country* will be returned. The query will attempt to match the triples in the graph pattern to a given model. Every binding becomes a query solution. It is important to note that SPARQL has a “property orientation” feature, where only class-attributes and properties can be matched to the query triples. For a good tutorial on SPARQL, we recommend (McCarthy 2005).

5.1.2 Metaweb Query Language (MQL)

MQL (Freebase 2007) is a recently developed query language for the open knowledge product from Metaweb, Freebase (Bollacker, Evans et al. 2008). It is built on top of the JSON protocol and it returns JSON objects. The language itself was created to allow online access to the Freebase Web Service. It is an effective approach to frame based querying using JSON notation and is quickly gaining traction. It does not easily allow transitivity operations and there is no notation for variable specification, but the simplicity of MQL is suitable for fast transactions and direct information access. It does not inherently contain a native inference mechanism.

```
{
  "type" : "/music/artist",
  "name" : "The Police",
  "album" : []
}
```

Figure 11 - MQL Example

Figure 11 shows a simple MQL query. In this case we are querying for all the albums from an artist with the name “The Police”. The result for the query described in Figure 11 would be something like Figure 12. Basically the idea is to complete the missing information expressed in the query.

```
{
  "type": "/music/artist",
  "name": "The Police",
  "album": [
    "Outlandos d'Amour",
    "Reggatta de Blanc",
    "Zenyatta Mondatta",
    "Ghost in the Machine",
    "Synchronicity",
  ]
}
```

Figure 12 - MQL Result Example

One important issue with using MQL in the federated ontology search framework is that MQL requires the user to be familiar with the structure of freebase, since it requires specific information in the query. For example, when specifying the *type* the query must be precise or no type will be matched, which means that the user must be familiar with the types available in Freebase. Given that in the federated approach the source ontologies can have an unlimited number of possible structures which the user cannot be expected to be familiar with, MQL in its current form is ill suited for a federated approach.

5.1.3 User Centric Approach to Ontology Query Languages

The main problem with the current available languages for ontology querying in a federated approach is the presupposition that the user has a priori knowledge of the structure of the ontology. This represents a data centric approach to querying where the user is asked to formulate the query in terms of the data contained in the ontology source rather than his or her information needs. In order to achieve independence from any one

specific ontology structure the query language in a federated approach has to be based on what is constant, independently of which ontologies will ultimately be queried.

One solution would be to normalize all ontologies into one specific ontological language. This approach has been attempted several times throughout the years [cite], always to no avail. Most large ontologies currently in existence are written in a specific format, whose authors argue, most times justly, has advantages over all other languages, be it in the inference mechanism associated with the ontology and native to the particular language or any other feature such as expressiveness, simplicity or efficiency. Also it is important to note that those resources already exist and thus it would be costly to convert them into a different representation. A federated approach should be agnostic to the underlying representation of the ontologies it queries, as far as the user is concerned.

Furthermore, we propose that the query language must focus on the user information needs rather than the structure of the available data. Thus we propose a language that is *User Centric* rather than *Data Centric*, that is, the query language models the information needs of the users, independently of the data structures available punting the burden of structure matching to the ontology search engine.

Let's take the following example. The user wishes to know the possible relationships between two concepts, concept *A* and concept *B*.

In SPARQL we would have to specify the relations that we would like to query for, the specific ontology that we were querying, etc. It is not possible to have something like Figure 13, where we would get all the relations between A and B.

```
PREFIX pref: <http://example.com/exampleOntology#>
SELECT ?x WHERE {
  A pref:?x B ;
}
```

Figure 13 - SPARQL Impossible Example

In MQL the problem is less acute, it is possible to get all the links between A and B, but only direct links. Exceptions are made in specific cases, where it is possible to devise highly customized queries that use nested queries. In Figure 14 we show a transitive query for locations. In this case the **id /guid/9202a8c04000641f8000000000959f60** represents the concept *US*.

```
{
  "query":
    [{"/location/location/containedby":
      [{"/location/location/containedby":
        [{"/location/location/containedby":
          [{"id":"/guid/9202a8c04000641f8000000000959f60"}]},
          "name":null}},
        "name":null}},
      "name":null,
      "type":"/education/university"}]
}
```

Figure 14 - MQL Transitive Query

The data centric approach forces the user to construct specific queries for specific languages forcing the user to adapt his or hers information needs to the data structure available.

Ideally we would like to be able to use a simple query like this

rel(A,B)

Figure 15 - Simple relation Query

Where the user is stating his or hers information needs by using operators that have a semantic meaning. In this case, the user does not care about the particular underlying structure but rather the goal of the query, that is, the information need.

Furthermore, it is important for any query language to allow adequate expression power while maintaining simplicity.

In order to achieve this we start with the desirable properties of a query language for a federated approach.

- Simplicity
 - A Query language should be simple such that the occasional user can use it for simple tasks without a big overhead in learning the language.
- Compositionality
 - Compositionality allows for the creation of complex queries based on simple operators. It gives the language flexibility to be used both by beginner and advanced users.

We approach this problem by proposing an operator based query language, inspired by the INQUERY query language (Callan, Croft et al. 1992). INQUERY was successfully used in a federated approach to IR (Si and Callan 2005) and is based on the same principles that we propose for a Federated Ontology Search approach. We will now describe a simple language for a User Centric ontology querying approach.

5.2 *Proposed Query Approach*

The success of the proposed approach hinges on the definition of a search method that is independent of any ontology. For this purpose we introduce the concept of *operator* and a concept of *query* based on operators. The main purpose of an operator is to decouple the search process from the information need. Instead of describing a complete semantic framework, the goal is to describe the information request in terms of a decomposable query that can be transformed into a set of operators. This would provide an elegant abstraction from the formal representations implemented by our ontological sources, allowing each operator to be an independent request.

It is important to note that by defining a set of operators we are in fact delegating responsibility for their execution to the ontologies themselves, therefore making no restrictions on whatever processes are executed in order to obtain the necessary information. This means that operators can be implemented using extended features of ontologies (e.g. inference, grounding, restrictions and theorem-provers). The only constraint is that the output of each query execution is a Directed Acyclic Graph (DAG).

5.2.1 Operators

An atomic operator is an atomic search operation on an ontology. It takes as input a graph and produces a ranked list of graphs as output. An operator is defined as an operation on

$$\text{op}(g): g \rightarrow g', \text{ where } g, g' \text{ is a DAG or}$$

$$\text{op}(g): g \rightarrow [g_1' \dots g_x'], \text{ where } g, g_n' \text{ is a DAG}$$

That is, an operator takes a graph g as input and return either g' or a list of graphs $[g_1' \dots g_x']$ as a result.

In this framework we divide the operator set into three operator types: Primary, Secondary and Boolean Operators.

Primary Operators form the core of the query procedure. They represent the basic operations that are typically performed in an ontology. They are fairly flexible and customizable. Secondary operators represent in fact a type of meta operators. They are specific configurations of the primary operators written in direct form for simplicity. Boolean operators perform the basic “*and*” and “*or*” operations in order to customize the query. Below we show the operator set followed by a description of the operators.

Operator Name	Operator Format	Operator Description	Example
Primary Operators			
Search operator	#search(S, opt)	Search takes a string	#search(“car”)

		and return a list of graphs that represent the possible nodes that are represented by that string	
Relation Operator	#rel(A, B, rels, opt)	Relation takes two concepts and returns the graph list that represent different paths from A to B	#rel(audi,machine)
Secondary Operators			
Concurrent Operator	#cont([B1..Bn], rels, opt)	Concurrent takes one or more concepts and finds concepts that are related with those concepts.	#conc([cough, fever, headache], symptom_of)
Define Operator	#define(X)	Define returns the graph representing the neighborhood of the concept X	#define(car)
Children Operator	#children(X)	Children returns the graph representing the children of concept X	#children(car)
Parents Operator	#parents(X)	Parents returns the graph representing the parents of	#parents(car)

		concept X	
Boolean Operators			
And Operator	#and(X,Y)	And returns the intersection of the two graphs	#and(#children(car), #children(plane))

The first thing one might notice when looking at the operator table is that we have only two primary operators. We will see how we can express all the other operators as a modified version of the relation operator. This is very good news. It actually means that in the worst case all we need is for an ontology to implement the two primary operators and we get all the other operators automatically implemented. In most cases there is benefit in implementing some of the other operators directly, but it is not a requirement. By reducing the number of operators we are actually alleviating the amount of work required to incorporate a new ontology into the system.

5.2.1.1 Primary Operators

There are two Primary operators, the *search* operator and the *relation* operator. The *relation* operator is mostly focused on finding relationships between concepts. Below we explain each operator in detail.

5.2.1.1.1 The search operator

The goal of the *search* operator is to return the available nodes that could correspond to a string. It takes optional arguments in order to contextualize the string. The *search* operator is denoted by the following general form

$$\#search(S, \{\text{optional arguments}\})$$

The goal of the search operator is to make the correspondence between strings and concepts. It is typically the first operation a user performs on an ontology, and is an exception to the rule, given that the input is a string rather than a graph.

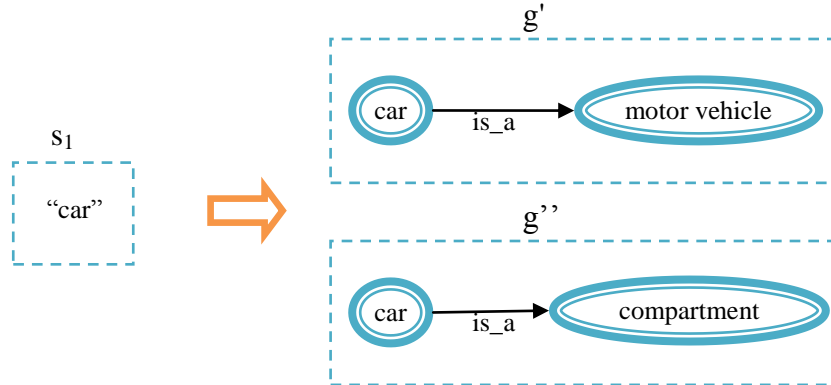


Figure 16 - Search Operator Example

The search operator takes as optional arguments a set of strings, whose purpose is the disambiguation of the possible concepts. That is, the context serves as a constraint on the matching between strings and concepts. Table 9 describes the optional arguments for the search operator.

Table 9 - Search Operator Optional Arguments

Optional Argument	Possible Values	Description
context	A set of Strings	Strings that are used to disambiguate the different meanings.

5.2.1.1.2 The relation operator

The relation *operator* is the most important operator of the operator set, since all the secondary operators are derived from it. The goal of the *relation* operator is to find relationships between two or more concepts. It takes relation constraints and optional arguments in order to constraint the operator.

The *relation* operator is denoted by the following general form

$$\#rel([A_1, \dots, A_x], B, [\text{relation constraints}], \{\text{optional arguments}\})$$

Figure 17 - relation operator general form

The relation operator takes as arguments a set $S = [A_1, \dots, A_x]$ and a concept B, a set of relation constraints and a set of optional arguments. The constraints are optional and default to unconstrained values. Table 10 specifies the optional arguments the operator can take.

Table 10 - Optional Arguments for relation Operator

Optional Argument	Possible Values	Description
Relation Constraints	A set of String values in the form $[R_1, R_2 \dots R_n]$	The relations the operator should restrict itself to. This can be interpreted as the restrictions on the paths the operator should use to propagate itself.
Context_A	Any set of string values	A set of strings that define concepts that might surround the concept A.
Context_B	Any set of string values	A set of strings that define concepts that might surround the concept B.
Edges	All Out In	Determines if the operator should use all edges, outgoing edges or incoming edges.
Expand	1..N	Determines how far the operator should propagate.

Figure 18 is an example of the relation operator in terms of graph representation.

Example : `#rel([car], vehicle,[is_a])`

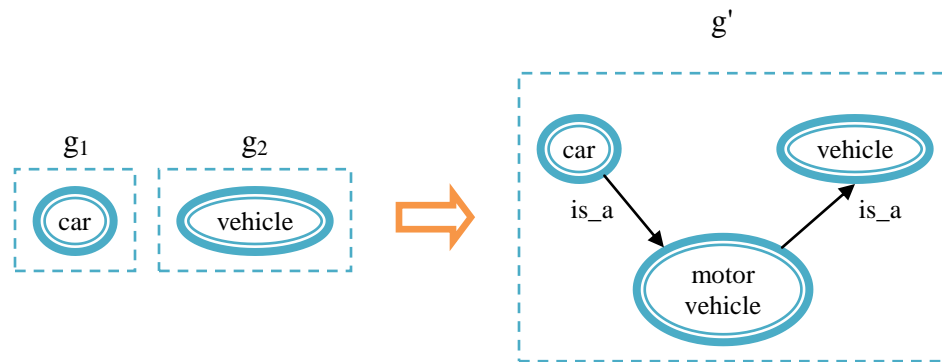


Figure 18 – relation operator

One important feature of the operator set is the use of null as a possible variable. In the case of a null value, the operator finds possible bindings to the variable denoted by the null value.

This principle is found in query languages such as MQL and Programming languages such as Lisp and Prolog, but when applied to the query operators it allows a new level of flexibility. The argument can be made that in fact all operations in an ontology are a version of one of two things: Finding a relation or finding a concept. One might counter argue that concept properties are modeled many times as something different than a concept, but in federated search a concept property is node in a graph with a relation to the concept, therefore the operator will return relations and nodes, be those nodes concepts or properties. By generalizing the representation we lose some expression power, namely the ability to restrict queries to properties, but we gain in flexibility.

In the case of *null* arguments consider the following question, “*what are the different types of car?*”

The query could be expressed as follows: `#rel([car],null,[type],edges=out)`. This could also be read as “*find all the concepts that are the descendents of car using the relation type*”. By having the second argument as null we are able not only find the paths from car to the second argument, but also consider multiple concepts as the second argument.

Later we will see the correspondence between secondary operators and the relation operator, but for now will give examples of how we can translate different questions into relation queries. As we can see in Table 11 the *relation* operator accounts for several different question types.

Table 11 - Example of the relation operator

Question	Query
What is the relation between car and machine?	#rel([car],machine)
Is car and machine connected by a type relation?	#rel([car],machine,[type],edges=out)
What are the different types of car?	#rel([car],null,[type],edges=out)
What are the parents of car?	#rel([car],null[type],edges=in)
What are the possible diseases have the symptoms of fever and headache?	#rel([fever,headache],null,[symptom_of])
What concepts are directed related to car?	#rel([car],null, expand=1)

5.2.1.2 Secondary Operators

5.2.1.2.1 Concurrent Operator

The goal of the *concurrent* operator is to find nodes that are concurrently related with a set of nodes. The concurrent operator tries to answer questions of the type “Given a set of nodes, what are the nodes related to this set?” One example of this question is the question “Given a set of symptoms, which diseases could the person have?”. The concurrent operator has the following general form.

#conc([Concept Set], {Relation Constraints},{Optional Arguments})

Figure 19 - Concurrent Operator General Form

The *concurrent* operator takes as arguments a concept set, a set of relation constraints and some optional arguments.

Table 12 - Concurrent Operator

Optional Argument	Possible Values	Description
Relation Constraints	A set of String values in the form [R ₁ ,R ₂ ...R _n]	The relations the operator should restrict itself to. This can be interpreted as the restrictions on the paths the operator should use to propagate itself.
Context_A	Any set of string values	A set of strings that define concepts that might surround the concept A.
Context_B	Any set of string values	A set of strings that define concepts that might surround the concept B.
Edges	All Out In	Determines if the operator should use all edges, outgoing edges or incoming edges.
Expand	1..N	Determines how far the operator should propagate.

The concurrent operator can be expressed in terms of the relation operator in the following way.

$$\#conc([\text{Concept Set}], \{\text{Relation Constraints}\}, \{\text{Optional Arguments}\}) = \#rel([\text{Concept Set}], \text{null}, \{\text{Relation Constraints}\}, \{\text{Optional Arguments}\})$$

Figure 20 - Concurrent Operator expressed in terms of the relation operator

5.2.1.2.2 Define Operator

The *define* operator's goal is to define a concept based on the node to which that concept is related. Basically the *define* operator is a convenience method to the following form of the *relation* operator:

```
#define(A) = #rel(A,null,{expand=1})
```

Figure 21 - The define operator as a formulation of the relation operator

The define operator makes it trivial to get a set of relations and concepts that define the concept.

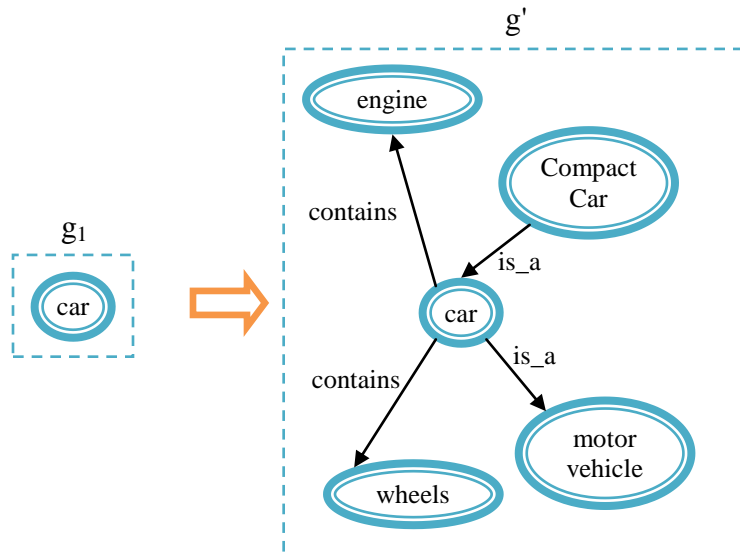


Figure 22 - Define Operator

Figure 22 is an example of the *define* operator. In this case we are depicting the definition of *car* as a set of relations to some possible concepts that are associated with the concept *car*.

5.2.1.2.3 Children operator

The children operator takes a graph g as input and expands each concept in g to the set of children concepts. Let's look at an example where the set of vertices of g , $V(g) = 1$, that is, the graph is comprised of one concept.

Query : #children(g)

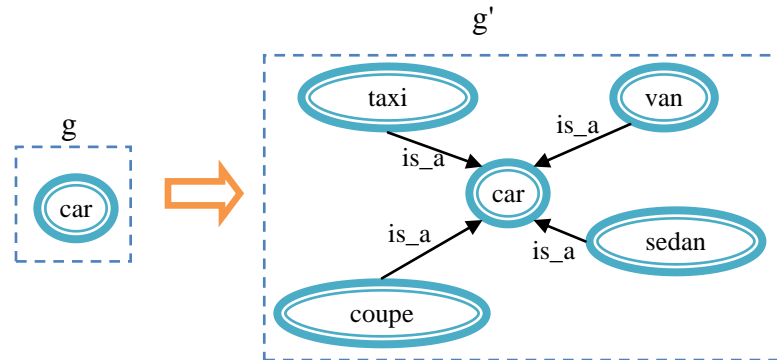


Figure 23 – Base case for children operator

The *children* operator can be defined as a specific formulation of the *relation* operator, given by

$$\#children(A) = \#rel(A, null, [is_a], \{ edges=out, expand=1 \})$$

Figure 24 - The Children operator as a function of the relation operator

As we can see in Figure 24, the *children* operator can be expressed as the *relation* operator where the target concept is left *null*, the expansion is limited to one level, the relations are constrained to *is_a* relations and only the outgoing edges are considered.

5.2.1.2.4 Parents Operator

The *parents* operator takes a graph g as input and expands each concept in g to the set of parent concepts. Let's look at an example where the set of vertices of g , $V(g) = 1$, that is, the graph is comprised of one concept.

Query : #parents(g)

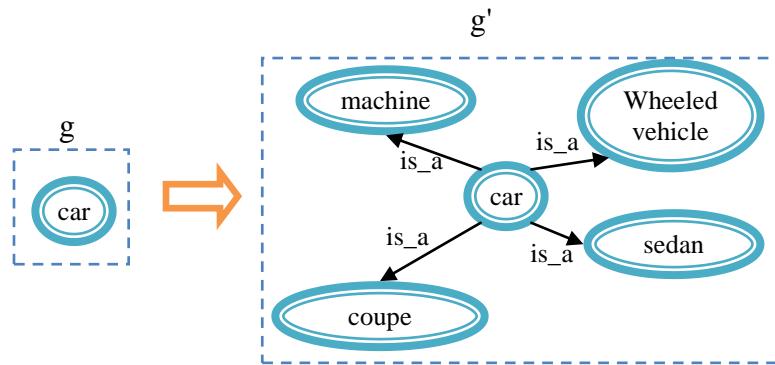


Figure 25 – Base case for parents operator

The *parents* operator can be defined as a specific formulation of the *relation* operator, given by

$$\#parents(A) = \#rel(A, null, [is_a], \{ edges=in, expand=1 \})$$

Figure 26 - The parents operator as a function of the relation operator

As we can see in Figure 26 , the *parents* operator can be expressed as the *relation* operator where the target concept is left *null*, the expansion is limited to one level, the relations are constrained to *is_a* relations and only the incoming edges are considered.

5.2.1.3 Boolean Operators

5.2.1.3.1 and operator

$$\#and(g_1, g_2)$$

Figure 27 - The and operator

The *and* operator represents the intersection operation. It takes g_1 and g_2 and return g' where g' is the intersection of the two.

Example

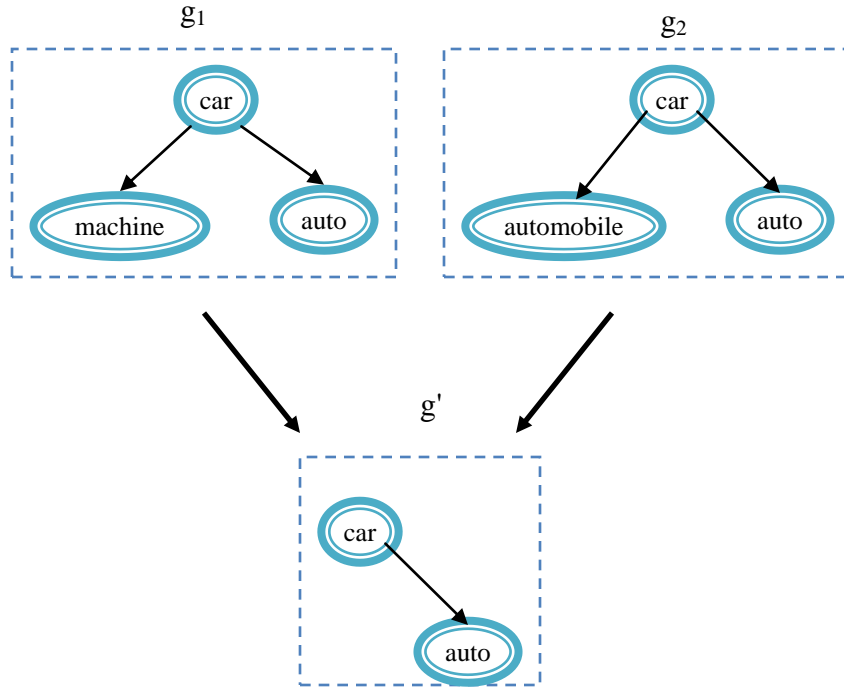


Figure 28 – and operator

5.2.1.3.2 or operator



Figure 29 - The or operator

The *or* operator represents the intersection operation. It takes g_1 and g_2 and return g' where g' is the union of the g_1 and g_2 . This operator does not apply boosting. Given two similar edges one of the edge is picked arbitrarily.

Example

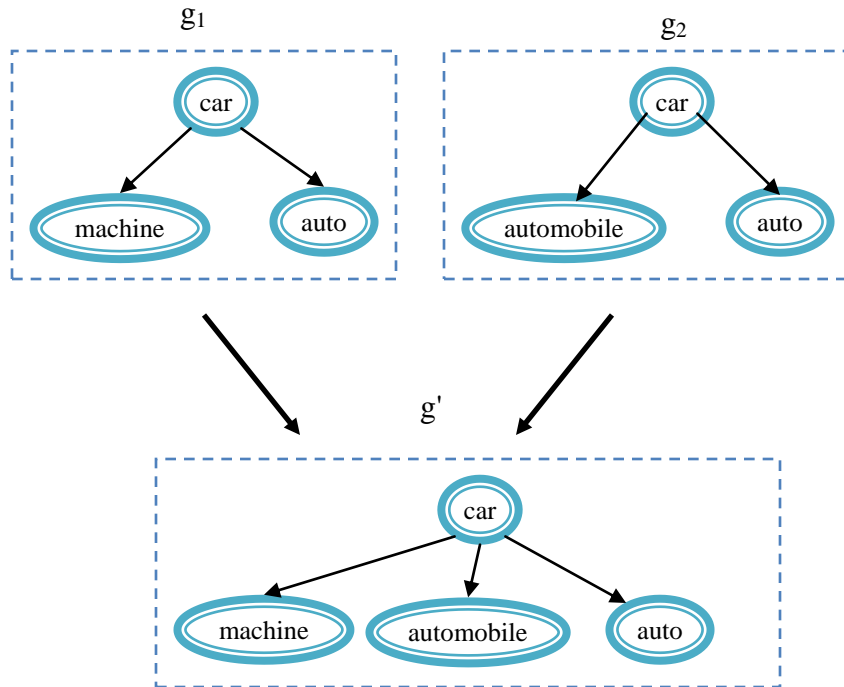


Figure 30 – or operator

5.2.2 Query

A query operation is composed of atomic operators and Boolean operators. Each query is reduced to a linear sequence of atomic operators.

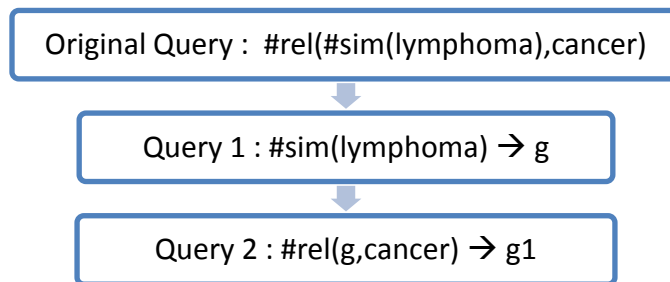


Figure 31 - Decomposition of operators

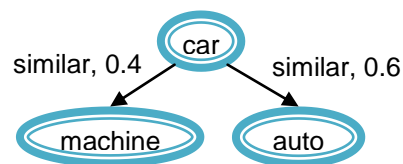
In Figure 31 we can see an example of operator decomposition. The property of compositionality allows us to create complex queries based on simple operators. This is extremely important but at the same time presents a problem of state explosion.

As the number of basic operators accumulates in a complex operator, the number of both the results and the concepts inside each result has a tendency to grow, rapidly becoming intractable. Therefore any implementation of this framework must take care to avoid this state explosion problem, by pruning query results to a limit of relations and operators. This will have an impact in accuracy, but is essential to make any system function efficient.

5.2.3 Query Results

A query result is a Directed Acyclic Graph (DAG). The graph contains labeled edges and concepts and attributes are modeled as nodes. Each edge contains a confidence associated with it. This confidence expresses the confidence of the source in the relation.

Example



The intuition behind this is that even though typically ontologies model knowledge in a binary fashion, much of the knowledge that is modeled by ontologies has a probabilistic nature. Let's look at the medical domain for an example. If we were to ask the question, **“what is the best treatment for a torn ligament?”** we might base our answer on the probability of success given the conditions of the patient. It might be **“surgery”** or **“physical therapy”**. But in both cases, we would be able to assign a confidence to each treatment that reflects the probability of the treatment working. Similarly, if we were to look at types of car we might be able to say that the most frequent type of is a sedan, given the number of cars sold worldwide. In this case, the confidence reflects which is the most frequent type of car. This type of information, even though many times not encoded in an ontology, is very important in federated search, since it plays a crucial role in the merging procedure. It is the job of the application layer to take the results of the queries, e.g. the graphs and interpret them in the appropriate fashion at a later stage.

5.3 Summary

Querying in the federated ontology search approach is proposed as a set of composite search operators, designed to describe the user needs rather than the data. It is the responsibility of the system to interpret the operators correctly and to apply them to each ontology. The operators are divided into primary and secondary operators. The secondary operators can be expressed by a specific configuration of the primary operators.

In this chapter we looked at ontological search. We described some languages used for ontological search, proposed a set of ontological query operators that take a user centric approach to ontological querying rather than a data centric approach. We discussed some of the shortcomings of the compositionally of the operators and described the query approach for Federated Ontology Search.

6 Result Merging and Scoring

This chapter describes and discusses the process of result merging and scoring. The goal of merging results is twofold. The first goal is to eliminate redundancy in the results when we have repeated results. The second goal is to use corroborating results to boost the confidence of the resulting results. We will first define the problem, talk about semantic distinction of results, describe concept normalization using synonym resources followed by issues in edge normalization and finally describe the merging algorithm and give some examples cases of merging.

As we discussed in the previous chapter, for every search query on the selected ontologies, one or more results is produced. Before we can produce a final set of results, we must maximize the utility of each result by merging the results whenever appropriate. The goal is to provide results that contain the combined consistent knowledge expressed in the union of the selected ontologies, thus making merging a crucial step in the process. Below we give an overview and contextualization of the result merging process.

6.1 Problem Definition

Let $O = \{o_1 \dots o_n\}$ be the set of available ontologies. Let q be the query performed. For each q we have a set of initial results R_i , such that $R_i = \{R_{o_1} \dots R_{o_n}\}$, where R is the result of applying function f to ontology o_x , or $R_{o_x} = f(o_x)$ and $f(o_x) = \{r_{o_{x_1}} \dots r_{o_{x_n}}\}$, where r_x is a Rooted Directed Acyclic Graph (RDAG). That is, given the set of available ontologies and a query, each ontology produces a set of results, thus each query produces an initial result set that is a set of sets of results. Merging can be described a transformation θ on the initial result R_i into the final result set R_f , or $R_f = \theta(R_i)$, where $R_f = \{r_1 \dots r_n\}, r_1 \approx r_n$. That is, the goal of Merging is to transform the initial result set, which is a set of result sets, into the final result set, which is a set of semantically distinct results.

In order to do this we follow the procedure of first defining equivalent and distinct results, merging equivalent result pairs until we are left with a set of distinct results.

6.2 Semantic Distinction of Results

Within a set of results, all the results were generated from the same query. This is an important point because it means that all the results are anchored either in the same concept or homonym concepts, or the root of the graph. Therefore, semantic distinct results are determined by the semantic distinction of the root. The fact that the root acts as a determiner for semantic equivalent results increases the efficiency of the result similarity algorithms, since we can reduce the search to the root of the graph and the children of the root.

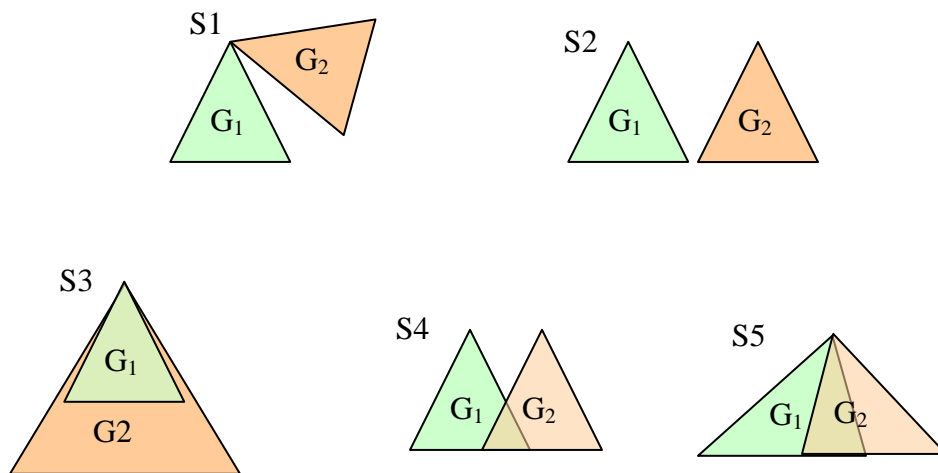


Figure 32 - Scenarios for Result Merging

Figure 32 illustrates the possible scenarios we can face when merging two results. *S1*, *S3* and *S5* represent situations with the same root concept. *S2* and *S4* represent situations where the root concepts are different. Scenarios *S2* and *S4* represent two results in which the roots are homonyms. Even though in *S4*, *G1* and *G2* have some common overlapping, we do not wish to merge the two results since their root concepts are different, thus representing different result. Let's consider the example were we perform a query regarding the concept *conductor*. Let us assume that we have two initial results that refer to *conductor* as the person who leads a musical group and *conductor* as a substance that readily conducts electricity. In this case, even if there is some common sub graph

(Scenario *S4*), we should present to the user two unmerged results, since they refer to distinct interpretations of the concept *conductor*.

Scenarios *S3* and *S5* represent two results where the root concepts of the graphs refer to the same concept. These cases represent the classical situation where it is desirable to merge the results, since each result complements the other. In the case of *S3*, since *G2* contains *G1*, *G1* will serve mainly for corroboration purposes.

The hardest case is *S1*, where the root concepts are the same, but there is no other overlap in the results. This case is extremely difficult to distinguish from *S2*, since we do not have further information in the results that can easily be used for disambiguation. In these cases we take the cautious route and do not merge the two results.

Determining the similarity of the results root concept hinges therefore on determining the overlap of the concepts in the concepts immediately adjacent to the root concept. If we can find overlap between the adjacent nodes in both results, then this is a strong indicator that we have a case of synonyms rather than homonyms and thus we should proceed to the merging algorithm.

The first step in merging two results is to normalize the results in order to minimize semantic ambiguities. Given that a result is a Rooted Directed Labeled Acyclic Graph (RDLAG) we focus on Edge Label normalization and Concept Normalization.

In the following sections I will describe the approach for graph normalization followed by the merging procedure.

6.3 Concept Normalization Using Synonym Resources

Concept normalization refers to the process of identifying and normalizing the synonym concepts. Given that the concepts are represented as nodes in graphs, the process consists

of identifying pairs of nodes from different graphs and replacing all occurrences with the same concept name in both graphs.

The basic idea is to use the resources created in chapter 3 as well as all the available ontologies to determine synonyms and solve terminological mismatches. On one hand we are addressing the problem of synonym terms, but we must avoid the problem of normalizing homonym terms. In order to avoid this we must replace sub graphs rather than individual nodes. The general intuition is that by normalizing subsets of the results that promote structural similarity we hope to avoid the problem of homonym terms.

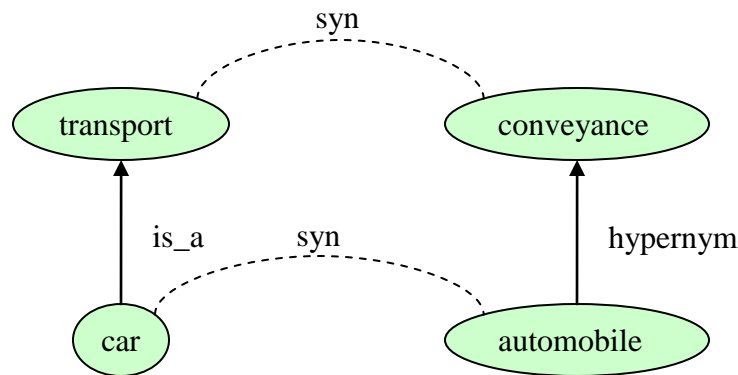


Figure 33 - Example of Concept Normalization

In Figure 33 we can see an example that illustrates the process. If *car* and *automobile* are synonyms, they will only be normalized if exists at least one other node to which they are both related. In this case we can see that *transport* and *conveyance* are also synonyms, so we perform a multiple node substitution. This prevents cases of homonym concepts, as demonstrated by the example below.

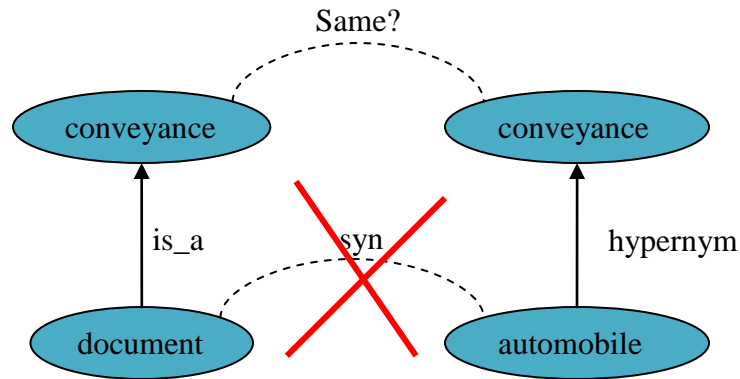


Figure 34 - Counter Example of Concept Normalization

As we can see in Figure 34, even though conveyance could be considered as the same concept, but since *document* is not a synonym of *automobile*, and we only replace sub graphs, we avoid replacing in this case. This heuristics eliminates most errors in homonym replacement, with the exception of homonyms sub graphs, which, even though it is possible, it is very rare due to the nature of concept organization.

More specifically Given $R_1 = \{C_1, E_1\}$ and $R_2 = \{C_2, E_2\}$ where R_1, R_2 are results, C_1, C_2 are the concept sets and E_1, E_2 are the edge sets and given $c_1 \in C_1, c_2 \in C_2$ where c_1 and c_2 are determined to be synonyms, we replace c_1 with c_2 if and only if $\exists c_x, c_y$ such that there is a relationship r_x connecting c_1 and c_x and a relationship r_y connecting c_2 and c_y , or $r_x(c_1) = c_x, r_y(c_2) = c_y$ and c_x and c_y are synonyms.

Normalizing the concepts is a two step process. First we identify replacement candidates using both the available synonym resources and a similarity measure such as the Q-Gram measure (Gravano, Ipeirotis et al. 2001). Finally we use the multiple node substitution to normalize the concept names. The pseudo code for the algorithm is given in Figure 35.

```
For each pair x, y, x is synonym of y
  For each pair w, z : w ≠ x, z ≠ y
    If w is connected with x and z is connected with y
      Replace x and w with y and z, respectively
    Exit for
  Else continue
```

Figure 35 – Pseudo code for concept normalization

Once we have normalized the concepts in both results we can then try to normalize the edge labels.

6.4 Edge Normalization

The relations found in the returned results represent edge labels and thus present a particular problem for graph normalization. Even though there is some consistency in traditional labels, such as *is-a* or *part-of*, a variety of relations exist that describe complex relations and typically consist of compound names that aggregate more than one word. In such cases we must find a way to normalize the edge labels by assigning the same labels to both results to be compared. One approach is to select the set of labels belonging to one result and apply them to the other result. The selected approach relies on the identification of synonym relation labels using a combination of methods, namely, direct synonym identification, string edit distance and structural similarity.

6.4.1 Label Synonym Identification

Similar to Section 6.3 we use the available resources created for synonym identification as the first step towards identification of synonym edge labels. Given graph G_1 and G_2 with edge label sets L_1 and L_2 respectively, we take each $l \in L_1$ and find the intersection between the synonyms generated by the synonym resources available and L_2 . This constitutes our initial candidate set. We extend our candidate set by calculating similarity between labels using a version of the edit distance measure, name the Q-Gram measure. For each pair of labels in the intersection we then compare the structures that contain the label pair. Given that the concept normalization was performed before hand, if we find overlapping structures, we choose one of the labels to normalize both labels. Given that we can have multiple candidate sets, it is possible that we generate more than one

possible substitution label set. Each substitution set consists of the labels in each result plus the substitution. The final label set is the one that results in the graph with the highest confidence.

6.5 Merging Procedure

6.5.1 Graph Similarity

Graph similarity Distance [A. Sanfeliu & K. Fu, 1981] is typically calculated in one of the following ways: Cost Based Distance, Feature Based Distance or Maximum Common Subgraph.

Cost Based Distance is based on edit operations on the graph, typically add nodes or edges, remove nodes or edges and re-label nodes or edges, where each operation is associated with a cost. Given two graphs g_1 and g_2 , the edit distance between g_1 and g_2 is the minimum number of edit operations necessary to transform g_1 into g_2 .

Feature based distances use a set of invariants established from the graph structural description, using these features in a vector representation to which we then apply distance or similarity measures.

The goal of the Maximum Common Subgraph approach is to find the largest Subgraph common to both g_1 and g_2 . To address this requirement, current approaches use the concept of maximum clique detection, or the concept of maximally connected sub graphs. Given the NP complete nature of the problem, the problem is then changed into finding the Maximum Common Edge Subgraph, which focuses on finding graphs with the maximum number of edges. In our case we use a variation of the overlapping coefficient for graphs, a measure whereby if graph g contains g' or the converse then the similarity coefficient is a full match.

6.5.1.1 Localized Boosting Algorithm

As stated before, Inexact Graph Matching is an NP complete problem. In order to tackle this problem, we take advantage of the fact that our graphs are RDAG's to reduce the complexity of the problem. The goal here is to create a set of tuples that will be the basis for comparison of the two graphs.

Given g_1 and g_2 as results of a query, the algorithm is as follows. After applying a screening procedure to determine the upper bound on similarity, as defined in (Raymond, Gardiner et al. 2002), we are left with graphs where $sim(g_1, g_2) > T$, that is, graphs where the similarity between g_1 and g_2 is above a certain threshold T , defined in the screening procedure. The screening procedure produces a subgraph for each graph given, which means that we now basically want to determine $g_1 \cap g_2$ for which we will apply localized boosting and then add the nodes and edges that were previously discarded. The final step is to resolve granularity differences in the graphs.

The basic intuition behind the confidence boosting is that the confidence of the edges is boosted whenever two edges are merged. The boosting is determined through the use of the *Soft Or*, given by the formula:

$$1 - \prod_i (1 - c_i)$$

The motivation of using *Soft Or* to determine the boosting is that this gives us a smooth boosting curve with an upper bound of 1.

E.g. $A = 0.8$, $B = 0.7$, $\text{Result} = 1 - (1 - 0.8)(1 - 0.7) = 1 - (0.2 \times 0.3) = 0.94$

In order to apply confidence boosting we apply the concept of tuples, where $t_x = (c_x, c_y, r)$ is a tuple, c_x, c_y are concepts and r is a relation.

First we split g_1 and g_2 into tuples $t_x = (c_x, c_y, r)$, $c_x, c_y, r \in g$, such that c_x and c_r are adjacent and $r(c_x, c_y)$. We then compare the sets of tuples from g_1 and g_2 and if $sim(t_x, t_y) > T$ then we boost the confidence of t_x .

An example is given in Figure 34, where two tuples are marked for merging and thus the confidence of the edge is boosted.

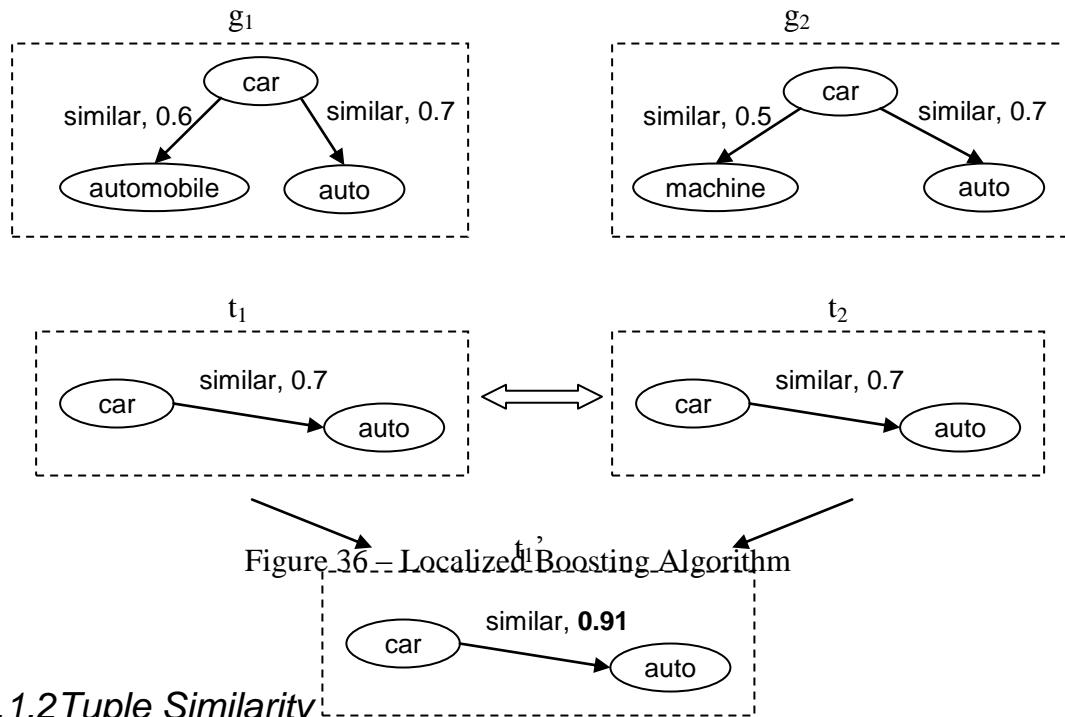


Figure 36. Localized Boosting Algorithm

6.5.1.2 Tuple Similarity

Tuple similarity measures are based on the linear combination of the edge similarity measure and the concept similarity measure.

When comparing concepts or relations, we use the Q-Gram distance on the strings that represent them (Gravano, Ipeirotis et al. 2001). A q-gram is character based N-Gram measure. The intuition behind the use of q-grams as a foundation for distance metric is that when two strings s_1 and s_2 are within a small edit distance of each other, they share a large number of q-grams in common. This metric is fairly robust to orthographic errors, morphological errors and compound words, which makes it suitable for our purposes.

The similarity between two tuples is given by the minimum similarity of the concepts and relations contained in the tuples, the intuition behind selecting the minimum similarity is basically to establish a conservative measure in which two tuples are only as similar as its most different part. If we look at different tuples, having different sources, target or relations makes them entirely different, therefore this relation requires that all parts be

similar. By using the minimum similarity we are saying that if either part is not similar then we will not merge the tuples. Formally

$$sim(t_x, t_y) = \min \begin{cases} sim(c_{x1}, c_{y1}) \\ sim(c_{x2}, c_{y2}) \\ sim(r_x, r_y) \end{cases}$$

Given that we already proceeded with the concept and edge normalization, determining similarity is basically a string comparison operation. The final step in the merging procedure is the Granularity Resolution, where we try to solve issues of different granularity in the two graphs.

6.5.2 Granularity Resolution

One of the biggest issues when merging two graphs is the problem of granularity resolution. Granularity problems stem basically from different granularity in the representation of the same information by two different ontologies and is one of the most interesting and unique problems to ontology merging.

Granularity issues typically describe a situation where the relation AyC in G_1 , denoting the relation between concept A and concept C , is described as AyB, AyC in G_2 . Figure 37 is a typical example.

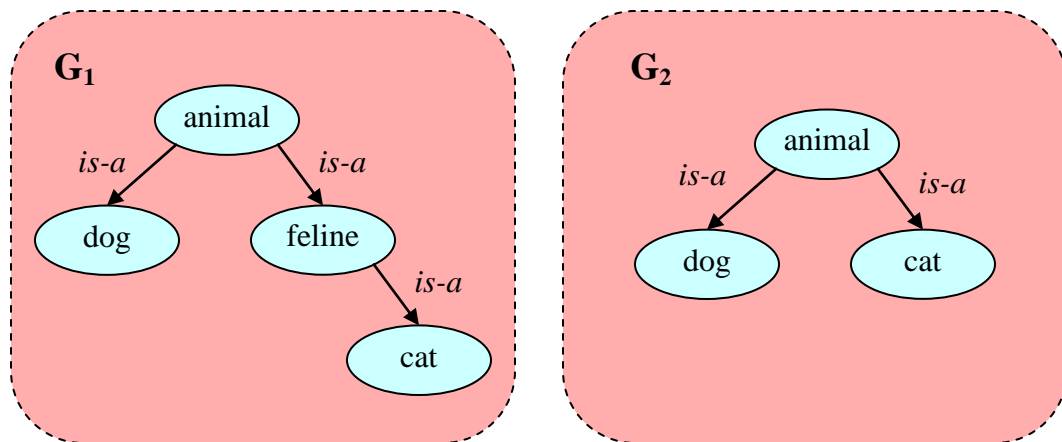


Figure 37 - Granularity Problem Example

In Figure 37 we can see two results that describe essentially the same information, but with different granularities. The relation $animal \rightarrow is-a \rightarrow dog$ remains consistent across results, giving us an the overlap required for merging, but while G_1 contains

$$animal \rightarrow is-a \rightarrow feline \rightarrow is-a \rightarrow cat$$

G_2 describes the relation as simply

$$animal \rightarrow is-a \rightarrow cat$$

Differences in granularity arise due to differences in the view of a domain, or simply due to the differences in the information need of the creators of the source ontologies. Perhaps the author of O1, from which G_1 is a result had the need to describe felines in a more detailed fashion than the author of O2, from where G_2 stems. Or perhaps O2 is describing the popular view of domestic animals. In either case both views must be considered without prejudice and assignment of correctness. The problem remains of how to solve the granularity issue when merging two graphs.

When solving the granularity issue, there are two factors we must take under consideration. First, the goal of Federated Ontology Search is not to generate the most correct ontologies or even to insure the plausibility of the results, but rather to retrieve and present information contained in the available ontologies, and do so in a best way to allow direct access to the requested information. Secondly, we have information associated with the ontologies and with each relation that allows us to compare both granularity views in respect to the overall confidence of the final result.

Therefore, we must merge the different granular paths in a way that maximizes the overall result confidence. Below is the example of what happens in the case of Figure 37.

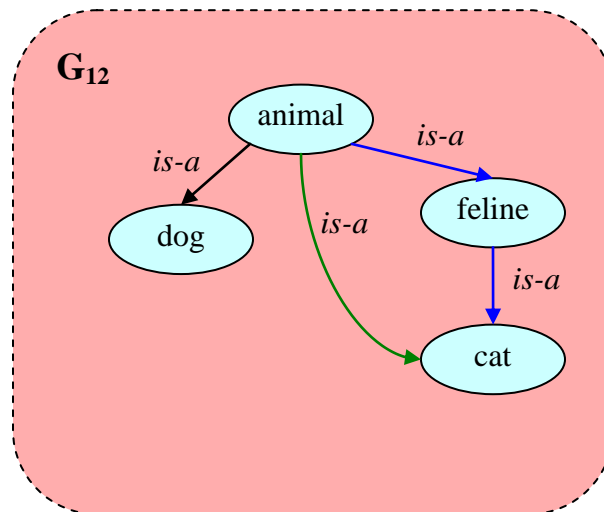


Figure 38 - Initial Merging

Initially the merging procedure merges all the nodes that do not overlap. After this initial step we find the nodes that have more than one incoming edge. These nodes represent the end nodes of structurally different paths. If the both incoming edges represent the same relation, then we use the Dijkstra algorithm (Fredman and Tarjan 1987) to calculate the best path to the source, the one with best confidence. Given that the final result score will be based on the confidence of the sources and the confidences of the relations in the results, we maximize the confidence of the result by maintaining the path with the best confidence.

The formula for the estimation of confidence is given by

$$\frac{\sum_{j=l}^r c(e_j)}{\sum_{j=l}^r e_j} \times C(O_s)$$

Where l is the leaf node, r is the root node, that is, the root of the divergent paths, C is the confidence and O_s is the source ontology. That is, the confidence is determined by average confidence multiplied by the confidence in the source ontology. After we determine which path contributes the most to the result score, we eliminate the weakest incoming edge.

Regarding the previous example, let's see what would happen given the following confidences table.

Source Concept	Target Concept	Confidence
Animal	Dog	0.8
Animal	Feline	0.7
Animal	Cat	0.8
Feline	Cat	0.6

Table 13 - Example Confidence Table

Source Ontology	Confidence
Ontology 1	0.8
Ontology 2	0.7

Table 14 - Example Source Ontology Confidence

$$\text{Confidence}(\text{animal} \rightarrow \text{is-a} \rightarrow \text{feline} \rightarrow \text{is-a} \rightarrow \text{cat}) = (0.7+0.6/2) * 0.8 = 0.52$$

$$\text{Confidence}(\text{animal} \rightarrow \text{is-a} \rightarrow \text{cat}) = 0.8 * 0.7 = 0.56$$

Thus the result is the following result since, all the paths represent the highest confidence from the root node to the leaves.

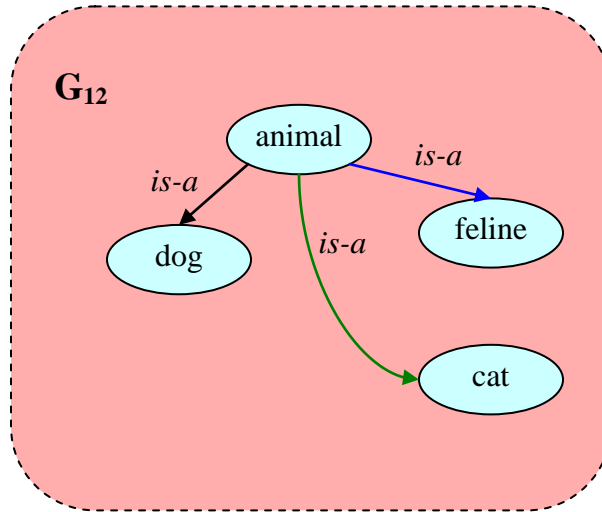


Figure 39 - Merging Result

One question that might arise is the argument that we are possibly losing information, since we no longer have the edge $feline \rightarrow is-a \rightarrow cat$, but I would argue that from the point of view of the user, the most reliable information is readily accessible. If this was the result of the query “*what is the relation between animal, cat and dog?*” the merged output would represent an appropriate result given the source ontologies.

Furthermore, one counter-intuitive result of the merging process is the possibility of having dangling non-leaf concepts. In the example presented previously, we can observe that the relation $animal \rightarrow is-a \rightarrow feline$ is maintained even though it doesn't have any more children. One could argue that non-leaf nodes should also be deleted, but in fact that depends of the semantic type of the relations. If instead of *is-a* relation we have a *synonym* relation, then all nodes are in fact potentially leaves. It is left for future work to determine what types of relations that can safely be removed whenever there are no child nodes left. This is an important issue because by leaving dangling leaves that were previously intermediate nodes we are creating structures that were inexistent in any of the existing ontologies. Even though this might not constitute a problem per se, changing structures carries unforeseen implications that need to be further studied.

The merging step leads to the final step in the search, the scoring algorithm, which we will discuss in the following section.

6.6 Scalability

The algorithms and procedures for matching results are mainly dependent on two factors, number of nodes and number of edges. For every pair of sub graphs we compare every tuple, which is comprised of two nodes and an edge. Since the nodes might be repeated, the best case scenario is the number of comparison is E , where E is the number of edges and the worst case scenario is $E + 2N - X$, where N is the number of nodes and X is the number of overlapping nodes in the tuples.

The algorithm is in the family of N^2 algorithms and as such it is not scalable to large graphs, but then the graphs are never quite big, and for good reason.

It is Important to note that the graphs returned by the ontologies are always fairly small. There are two main reasons to justify this. First, the fact that the queries are anchored in a specific concept or concept reduces the ambiguity and size of the returned graphs. Second because even when an ontology has a large average path size, which is not common, meaningful relations between concepts suggest a correlation to close concepts in the ontology. Also, for computational purposes, large graphs are not computable in real time, so it is computationally desirable to maintain the size small.

As the number of ontologies grow, the proactive ontology selection algorithm will provide the scalability necessary, since we can impose restrictions on the maximum number of ontologies to be queried, based on the order of relevancy of those ontologies for a particular query.

6.7 Result Scoring

The goal of result Scoring is to rank the results after the merging procedure. The more complete, direct and trustworthy a result is, the higher it should rank. The scoring metric should thus give preference to results that have a higher average span and shorter path lengths. Let's take the following simple examples.

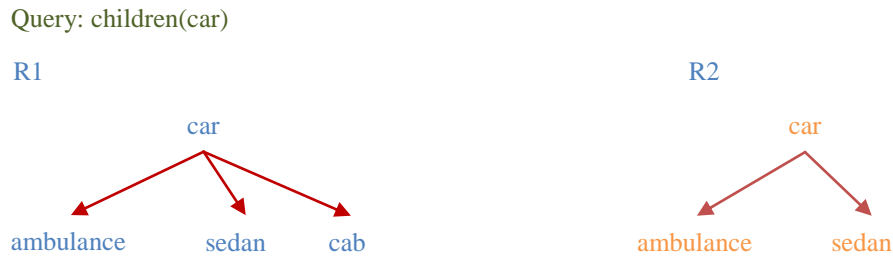


Figure 40 - Result scoring example 1

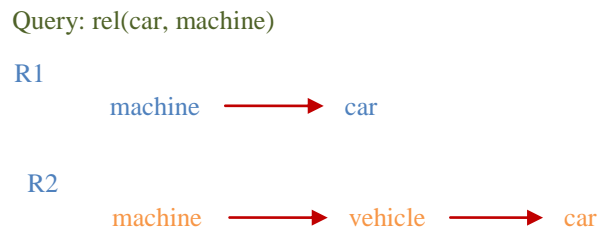


Figure 41 - Result scoring example 2



Figure 42 - Result scoring example 3

In the example 1, the result that should be scored the highest is the first, since it contains

more information. In the second example, result 1 had a direct connection between the source and the target of the query. We argue that given that the user query pertains to the relationship between car and machine, if we are able to find a direct connection between those two concepts, that forms a stronger bond than the result *R2*, thus scoring higher. Also, we posit that the correlation between the confidence of the edge and the confidence that the relation described by that edge satisfies the query decreases with the increase of the distance to the source node.

In example 3 we can see a query that tries to identify possible diseases that contain the symptoms of fever and headache. Again, result 1 should score higher since it contains more information. In all these examples we can observe the principles we outlined earlier. Larger average spans and shorter paths.

Thus we propose the following scoring metric that adequately models the desired behaviors:

$$C_r = C_o \times \frac{\sum_{i=1}^n \frac{C_{e_i}}{\varphi_{e_i}}}{|e|} \times \Delta_d$$

Where C_r is the confidence of the result, C_o is the confidence of the source, φ_{e_i} is the distance of the target of e_i to the source of the path, C_{e_i} is the confidence of the edge e_i , $|e|$ is the number of edges and Δ_d is the edge degree delta, described below.

$$\Delta_d = \frac{Avg Degree}{Max Avg Degree}$$

Where *Avg Degree* is the average degree of the result and *Max Avg Degree* is the maximal average degree from all the results being considered.

Let's observe what happens in example 2 if the following conditions. Let us assume that the source ontology for R1 and R2 have the same confidence of 1, that is, we assume that they are completely reliable in the information they contain, and that each edge has the same confidence of 1, that is, we also trust each edge completely. Even though these conditions are not typical, they serve the purpose of illustrating the scoring metric.

For *R1*, the score would be

$$C_{R1} = 1 \times \frac{\sum_{i=1}^1 1}{1} \times 1 = 1 \times 1 \times 1 = 1$$

For *R2*, the score would be

$$C_{R1} = 1 \times \frac{1 + \frac{1}{2}}{2} \times 1 = 1 \times 0.75 \times 1 = 0.75$$

So the final ranking would be:

Final Ranking
R1
R2

If we experiment with different confidences or different examples, we can see that the scoring outcomes follow the intuitions that guided our scoring principles. Even though the ultimate goal of the scoring metric is to provide an immediate value of how good the result satisfies the query, at this point we do not make such claims. We believe that correlation between the values and the quality of the results is not well enough established, thus making only use of the scoring metric for ranking purposes and leaving the quality correlation as future work.

The description of the scoring metric concludes the presentation of the thesis framework. In the next chapter we will discuss the results obtained during this thesis work, followed by conclusions.

7 Result Discussion

Evaluating Ontological based research has been traditionally a difficult task. Given the multitude of ontology languages and the lack of existing standard test sets for ontologies the most common approach to ontology evaluation is a task centered evaluation process (Porzel and Malaka, 2004). The goal of task based evaluation is to indirectly evaluate an ontology by the effect that the ontology has on the task performance. Though this method does not directly evaluate the quality of the ontology, it is debatable if that is even possible. An ontology is, after all, a view of a domain from the perspective of one or more people, and its structure and contents highly subjective. The attempt at evaluating the quality of the knowledge contained in an ontology often proves as subjective as the ontology itself. The method of task based evaluation provides us with clear metrics that, despite not perfect, allow us to judge the usefulness of a particular ontology in a particular task. This method is particularly suited for the evaluation of a Federated Approach, since the goal in this case is to compare the Federated Approach with a standard approach. By evaluating on a task based evaluation, we can acquire clear comparative data of the two approaches and the advantages and disadvantages for a Federated Approach.

In this chapter we focused on the evaluation at a broad level, where we are concerning ourselves mainly with the performance of the whole system. The evaluation of individual portions of the system, such as the automatic creation of ontologies, is presented in the chapters pertaining to the portion of the system in question.

This chapter is divided in two parts. First I will describe experiments performed in the area of question answering, where Federated Ontology Search was applied to the task of Type Checking, where the goal is to improve the performance of the question answering system by boosting the answer candidates that are of the correct type. Second I will describe the experiments performed in the area of content matching. Here I will describe the impact of Federated Ontology Search in three tasks; Concept Recognition, Concept Disambiguation and Concept Matching.

7.1 Type Checking

The task of type checking tries to determine if, given a type T and a concept C, C is of type T. In the case of the federated approach, we can achieve this by using three operators, the *relation* operator, the *parents* operator and the *children* operator, as previously described.

Type checking using federated ontology search can be viewed as the task of finding an *is-a* based path between two concepts. Our approach has the advantage of using indirect paths when no direct path is found. An indirect path consists of partial ordered sub paths that exist in separate ontologies but form one path when combined. Finding an indirect path is possible by simply applying either the *parents* operator or the *children* operator to the source node in one ontology and using the resulting nodes to query for a direct path in another ontology. The resulting path is the combination of these partial paths. Using indirect paths provides a promising way of combining information that by itself would be incomplete and enabling the deduction of previously non-existent paths.

7.1.1 Experimental Setup

A total of 9558 pairs were extracted from results of the Javelin question answering system in TREC QA 2003 [Nyberg et al., 2003]. Each pair consists of the expected answer type or subtype and the candidate answer.

For the purposes of our evaluation we used two of the currently available ontologies, Wordnet and ThoughtTreasure. The purpose of this preliminary evaluation is to contrast the performance of each of the ontologies individually, which would be a typical scenario for a project using one ontology as a knowledge base, with the performance of the set of ontologies using a federated approach.

We have evaluated the recall and precision of the retrieved results..

7.1.2 Results and Analysis

Table 1 shows the recall after running the test set with different configurations.

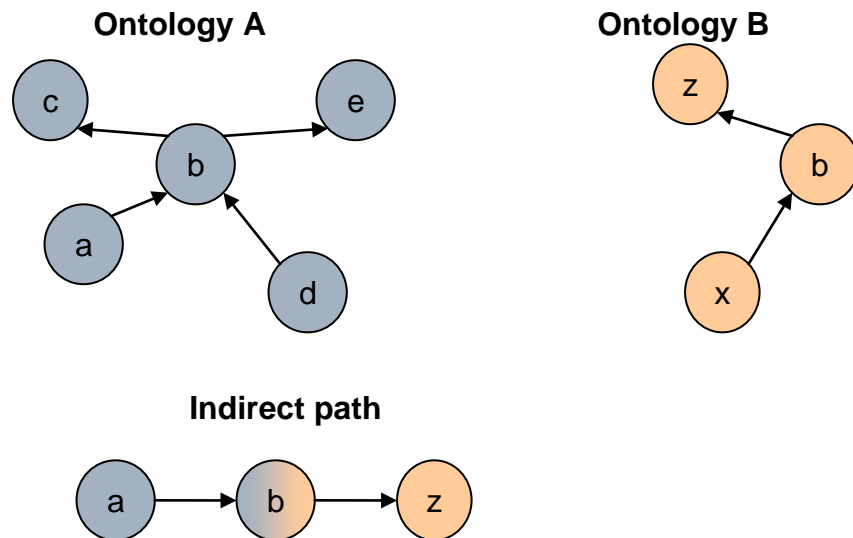
Configuration	Recall
---------------	--------

Wordnet	4278 (44.7%)
ThoughtTreasure	730 (7.6%)
Combined	4686 (49%)
Merge	4686 (49%)
Merging + Indirect	6870 (71.8%)
Test size	9558

Table 1: Recall using different configurations with the full set of pairs

Wordnet and *ThoughtTreasure* were experiments where Wordnet and ThoughtTreasure were used individually. The *Combined* experiment queried each of the ontologies individually, picking only the top ranked result. The recall is lower than the direct sum of the individual results due to knowledge overlap in the ontologies. The *Merge* experiment queries both ontologies but merges the results using the merging algorithm described previously. Finally we use merging as well as indirect path query to perform the last experiment

An indirect path is a path that is comprised of partial paths contained in different ontologies, as shown below.



Although the recall remained the same when applying the merging procedure, the average confidence of the top result, in cases where there was more than one result, increased significantly (28%), as shown in Table 2.

	avg. confidence
Without merging	0.72
With merging	0.93
Increase	28.7%

Table 2 – The Increase in the average confidence of the top ranked result due to the merging algorithm.

In order to test the accuracy of the federated approach, we created a gold standard for a subset of the full set of pairs. Using random sampling, we selected 1300 pairs, which we then proceeded to judge manually. For each pair in the gold standard subset we generated a tuple of the form $(type, concept, judgment)$, where judgment reflects if the *concept* is of the type *type*.

We compared the answers of the Federated Search with the gold standard by applying a variation on the result score threshold. If a score is below the threshold then the *concept* is considered not to be of the type *type*.

	Precision	Recall	F1 Measure
Combined (W+T)	0.59	0.49	0.53
FOS (M+I)	0.67	0.71	0.69
Increase			30.18%

Table 15 - Precision and recall of the Federated System using Wordnet and ThoughtTreasure

We obtained a significant increase in performance when using the federated search approach. The optimal threshold for this experiment is $T=0.1$ with a precision of $P =$

0.676. The recall was very close to the one obtained in the full set with a recall of 0.71 (71%). In Table 15 we can see the F-Measure of the system.

7.2 Content Matching

The goal of content matching in general is to determine the how well two content pieces fit together. Let's say for example that we have two images, Image A and Image B. The goal of the task, generally speaking, is to determine how well Image A and Image B match. In the context of this work this is restricted to content that contains a set of keywords that describe it. Thus, content matching is the task of matching two sets of keywords. This task has many practical applications, such as content recommendation, filtering, online ad matching, content personalization, etc. This task is also particularly suited to highlight the potential of the federated approach, since the keywords can contain any string, thus challenging the coverage of any ontology. As previously mentioned, there are currently no standard test sets suited for ontology testing. Part of the task is the definition of the task itself, the acquisition and labeling of data and the creation of the test set, which constitutes one of the deliverables of this thesis.

7.2.1 Task Definition

The task consists of the following. Given two objects, A and B, described each by a set of keywords $K_A = \{A_1, \dots, A_n\}$, $K_B = \{B_1, \dots, B_n\}$, the goal is to determine $S(A, B)$ which is the semantic distance between A and B. We want S to correlate with semantic distance such that the highest the score the closer the two sets are semantically. Given that we compare set A with sets B and C, if $S(A, B) > S(A, C)$ then it should be expected that A is closer to B than to C as judged by a person.

7.2.2 Experimental Setup

Our data comes from two sources. First we use the data collected by Louis Von Ahn (cite) with the ESP Game, that consists of images labeled with keywords, also known as tags. Our dataset consists of 50.000 images with the respective tags. We then take 10.000 randomly selected images and use the tags of each image to search for products in Amazon's product API. For each image we collect 10 products, each product containing

tags that were inserted by Amazon users. There is no restriction in either tag set. The products are ordered by relevancy regarding the image.

We then take each set comprised of the image and the 10 ordered results and label each relation in the form of *(Image, Product, Rank)* using Mechanical Turk. Each relation is labeled by three different people as either *relevant* or *not relevant*. In order to minimize cultural bias, only Mechanical Turk users from the United States were allowed to participate in the study. The final determination of relevance was determined by simple majority of the labels. That is, for every given relation, 0 or 1 labels of *relevant* will produce a final *not-relevant* label, 2 or 3 labels of *relevant* will produce a final label of *relevant*. Figure 43 shows an example of one of the Mechanical Turk task.

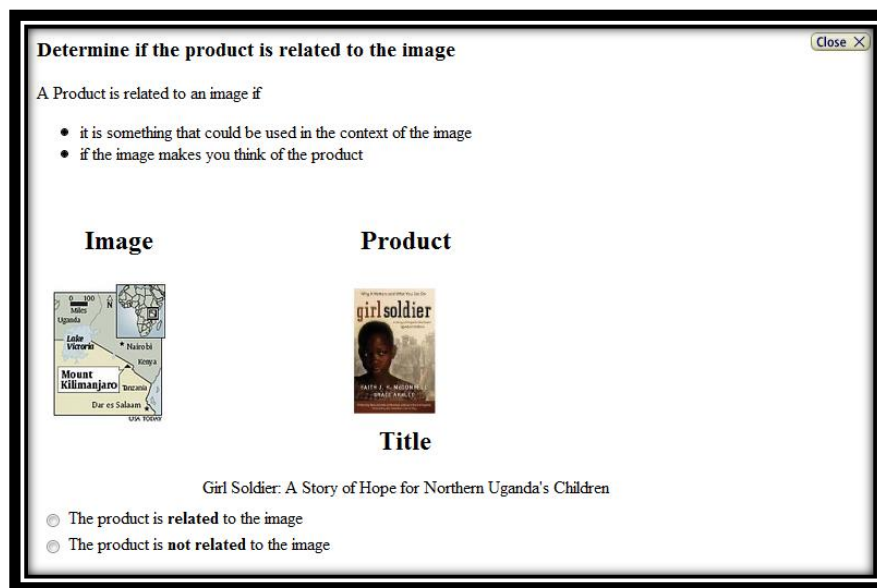



Figure 43 - Example of Mechanical Turk task

After data collection and labeling we have a set of tagged images, each with a set of ordered tagged products, and for each product a label of either *relevant* or *non-relevant*. Below is an example of the final result, which constitutes our gold standard. In this example, we have one image with 12 associated products. For each product we got 3 different people to judge whether the product is relevant to the image. The products are ranked from 1 to 12 and only products 11 and 12 were considered relevant, since all three

labelers thought they were relevant. Products 4, 9 and 10 were considered irrelevant even though 1 labeler out of 3 said the product was relevant.

Image	Tags	Rank	Product	Tags	Score	Relevance
	Season Collection Series Dvd Cd Show Tv Box Pink red	1		canning kit, canning, food preservation, canner, canning rack	0	No
		2		pressure canner, canner, canning equipment, pressure cooker, canning	0	No
		3		pressure canner, canning equipment, pressure cooker, built like a diving bell, canner	0	No
		4		david cook, american idol, analog heart, axiom, american idol winner	1	No
		5		classical music, joshua bell, music, baroque, chamber music	0	No
		6		jersey boys, broadway, four seasons, musical, broadway musical	0	No
		7		Disney, movie classics, vhs, cross reference, dick van dyke	0	No
		8		Boxing, Ireland, dowery, irish, john wayne	0	No

		9		childrens movies, dvd	1	No
		10		the office, steve carell, john krasinski, comedy, rainn wilson	1	No
		11		Heroes, superheroes, series, tv, dvd	3	Yes
		12		greys anatomy, tv series, katherine heigl, ellen pompeo, female main character	3	Yes

Table 16 - Example of Product Review

Below we present some results regarding the agreement of the users that reviewed the data in Mechanical Turk. Due to the lack of overlap of reviewers, given the distributed nature of Mechanical Turk, agreement cannot be calculated using a K measure. To alleviate this problem we describe the agreement in terms of the general measure of relevancy. That is we measure agreement at a local level, since each task was done by three different reviewers. In figure 44, unanimous represents when all reviewers agreed on the answer either positive or negative. Figure 45 and 46 show the distribution of relevance and agreement, respectively.

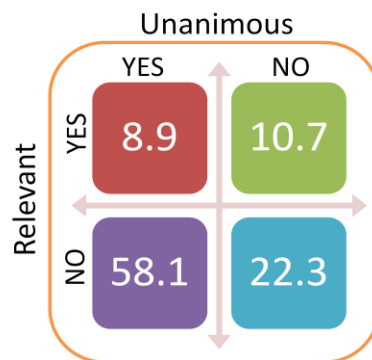


Figure 44 - Agreement between reviewers

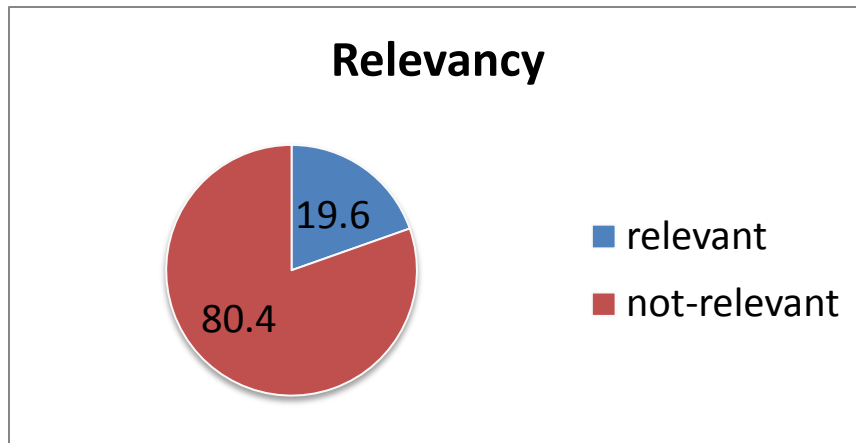


Figure 45 - relevancy of products

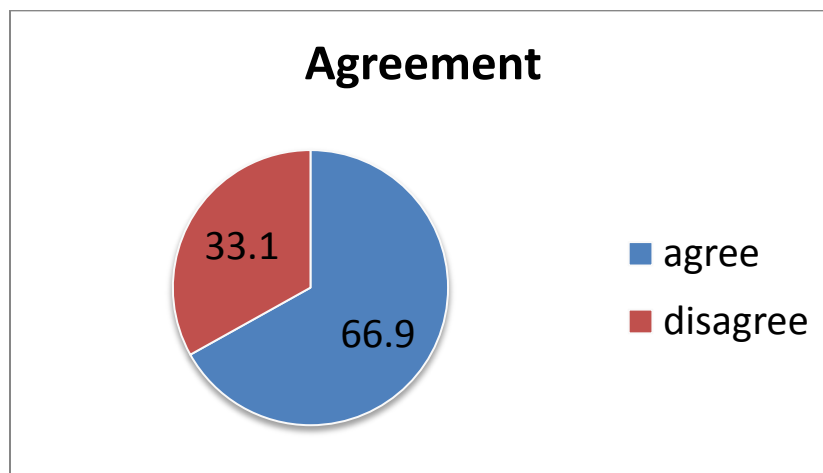
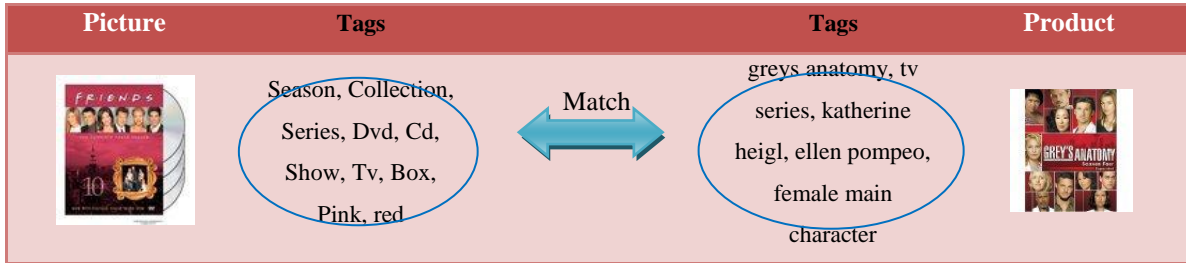


Figure 46 - Agreement between reviewers

8 Result Analysis

The matching procedure uses the FOS engine for all the steps requires to produce a matching score. The goal is to take two sets of keywords and return a relevancy score. In this particular task the keywords represent the tags associated with a particular content.

Table 17 - Example of Matching Inputs



It is important to note the subjectivity of this task. Relevance judgments are notoriously subjective to the people that make them and especially in this case, since the association is made with little information. There are many factors that can account for difference in judgments of the same thing by different people. Some of those factors, such as culture, were controlled by limiting the US persons participating in the task, but we acknowledge that it is impossible to account for all the factors and in fact we do not try to do so. The goal of the task is to try and capture the subjectivity of a task that humans perform on daily basis but the goal of the thesis is not to demonstrate improvements in relevance, even though we do in some cases, it is to demonstrate the effect of federated ontology search in the task. Thus we focus not on increasing accuracy, but rather in demonstrating the effect of our approach in both the main task and the subtasks contained in it.

In order to measure the results obtained in the subtasks performed, we used a set of ontologies comprised of Wordnet, Freebase, OpenCyc, UMLS, UMBEL and DBPedia incorporates within the Federated Ontology Search system. This ontology set provides an interesting combination of various ontology types. Wordnet and OpenCyc are carefully supervised ontologies, Freebase is user generated, DBPedia is automatically generated from Wikipedia and UMLS is a collection of dictionaries from the medical domain. We

measured our results by comparing the performance of the individual ontologies in each subtask with the performance of the system as a whole.

In order to obtain the final relevance result, we subdivide the task into three subtasks. Namely string coverage, concept disambiguation and concept set matching.

8.1.1.1.1 String Coverage

The first step in the matching process is to produce a set of possible concepts for each of the keywords. This is equivalent of using the *search* operator in the federated ontology search engine. We take each set of keywords and for each keyword we execute search operator. That is, for each keyword k_x in the keyword set $K = \{k_1, \dots, k_n\}$ we execute *search*(k_x). This produces a set of potential concepts of the form $C_{k_x} = \{c_1, \dots, c_n\}$. Table 18 exemplifies a search operation and the results.

Keyword	Car
Query	search(car)
Result	car(motor vehicle) car(wheeled vehicle) car(compartment) car(Greek mythology) car(Musical Track)

Table 18 - Search Operator Example

We can observe the effect of federated ontology search in this task by measuring the coverage within fundamentally different keywords sets. The keywords used in the images are of a more general nature, with the 3 most frequent words being red, blue and black. The keywords used in the Amazon products are much more specific since they refer to specific aspects of each product and thus create a bigger challenge for this task.

Below we present the results of comparing the performance of the FOS system and a set of baselines.

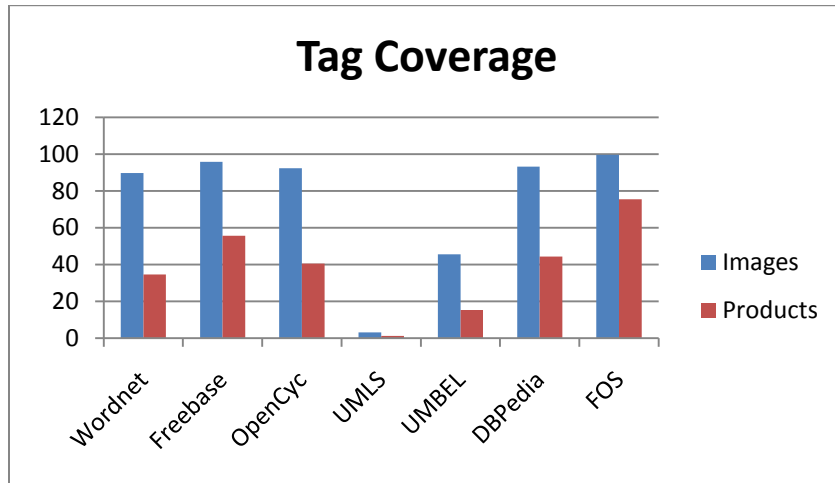


Figure 47 - Coverage Results

Coverage	Wordnet	Freebase	OpenCyc	UMLS	UMBEL	DBPedia	FOS
Images	89.75	95.74	92.23	3.12	45.56	93.2	99.6
Products	34.65	55.68	40.54	1.23	15.43	44.3	75.4

Table 19- Coverage Results Table

Statistical significant with $p < 0.0001$

When comparing the best baseline to using the FOS system, we can observe that Freebase had 95.74% coverage of the image tags and 55.68% of product tags coverage. The FOS system presented an improvement of 4% and 35% of coverage increase in the images and products respectively.


The results shown can be explained by the difference in nature from the tags used in the image to the tags used in the Amazon products. Even big ontologies such as Freebase contain a large portion of general use words in the English language, but are not particularly effective in covering specific nouns and especially in compound words such as **pressure canner**, **garment steamer** or **christian movie**. By adding ontologies such as DBPedia Wordnet and OpenCyc, the coverage of the system expands into concepts that used in specific environments by real users, allowing for a significant increase in coverage.

8.1.1.2 Concept Co-Disambiguation

Once we have the possible concepts for each string, we need to select the right concept. We do this by searching for relationships between concepts and using belief propagation (Murphy, Weiss et al. 1999) to find the most likely configuration. We disambiguate words based on a simple principle of cues. For humans, it is very easy to disambiguate *turkey* in (*turkey, Istanbul*) into the country turkey, versus *turkey* in (*turkey, thanksgiving*) into the bird turkey. Humans can do this because surrounding concepts allow them to disambiguate easily when taken in conjunction. We use the same principle by disambiguating strings in pairs and call this method co-disambiguation. For every pair of possible concepts within the tag set, we find the possible relationships for that pair. We then create a graph in which each tag is a random variable, where the concepts are the possible values for the variable. We then run loopy belief propagation to get the MAP assignment for the variables.

In disambiguating, we found that disambiguating to the wrong concept is more costly than not to disambiguate at all, since it will lead the matching procedure into the wrong path. Therefore we only disambiguate when there is enough confidence to so, which leads to a smaller number of disambiguated concepts. Figure 48 shows an example of the whole process.

For each set of tags we start by extracting all the possible concepts for each of the tags in the set. We then search for all the possible relationships between each of the concepts in different keywords. Afterwards we create the belief graph and finally we run loopy belief propagation in order to disambiguate the tags into concepts.

Image	 <p style="text-align: center;">results</p>					
Tags	Uganda	Kilimanjaro	Nairobi	Kenya	mount	victoria
Concepts	C ₁ →uganda →country C ₂ →uganda →song	C ₃ →kilimanjaro →peak C ₄ →kilimanjaro →album C ₅ →kilimanjaro →song	C ₆ →nairobi →capital C ₇ →nairobi →song	C ₈ →kenya →country C ₉ →kenya →song	C ₁₀ →mount →attach C ₁₁ →mount →increase C ₁₂ →mount →fix C ₁₃ →mount →horse C ₁₄ →mount →rise C ₁₅ →mount →elevation C ₁₆ →mount →music	C ₁₇ →victoria →queen C ₁₈ →victoria →deity C ₁₉ →victoria →lake C ₂₀ →victoria →city
Queries	rel(C ₁ ,C ₃) rel(C ₁ ,C ₄) rel(C ₁ ,C ₅) rel(C ₁ ,C ₆) rel(C ₁ ,C ₇) rel(C ₁ ,C ₈) ... rel(C ₂ ,C ₁₅) rel(C ₂ ,C ₁₆) rel(C ₂ ,C ₁₇) rel(C ₂ ,C ₁₈) rel(C ₂ ,C ₁₉) rel(C ₂ ,C ₂₀)	rel(C ₃ ,C ₆) rel(C ₃ ,C ₇) rel(C ₃ ,C ₈) rel(C ₃ ,C ₉) rel(C ₃ ,C ₁₀) rel(C ₃ ,C ₁₁) ... rel(C ₅ ,C ₁₅) rel(C ₅ ,C ₁₆) rel(C ₅ ,C ₁₇) rel(C ₅ ,C ₁₈) rel(C ₅ ,C ₁₉) rel(C ₅ ,C ₂₀)	rel(C ₆ ,C ₈) rel(C ₆ ,C ₉) rel(C ₆ ,C ₁₀) rel(C ₆ ,C ₁₁) rel(C ₆ ,C ₁₂) rel(C ₆ ,C ₁₃) ... rel(C ₇ ,C ₁₅) rel(C ₇ ,C ₁₆) rel(C ₇ ,C ₁₇) rel(C ₇ ,C ₁₈) rel(C ₇ ,C ₁₉) rel(C ₇ ,C ₂₀)	rel(C ₈ ,C ₁₀) rel(C ₈ ,C ₁₁) rel(C ₈ ,C ₁₂) rel(C ₈ ,C ₁₃) rel(C ₈ ,C ₁₄) rel(C ₈ ,C ₁₅) ... rel(C ₉ ,C ₁₅) rel(C ₉ ,C ₁₆) rel(C ₉ ,C ₁₇) rel(C ₉ ,C ₁₈) rel(C ₉ ,C ₁₉) rel(C ₉ ,C ₂₀)	rel(C ₁₀ ,C ₁₇) rel(C ₁₀ ,C ₁₈) rel(C ₁₀ ,C ₁₉) rel(C ₁₀ ,C ₂₀) ... rel(C ₁₆ ,C ₁₇) rel(C ₁₆ ,C ₁₈) rel(C ₁₆ ,C ₁₉) rel(C ₁₆ ,C ₂₀)	

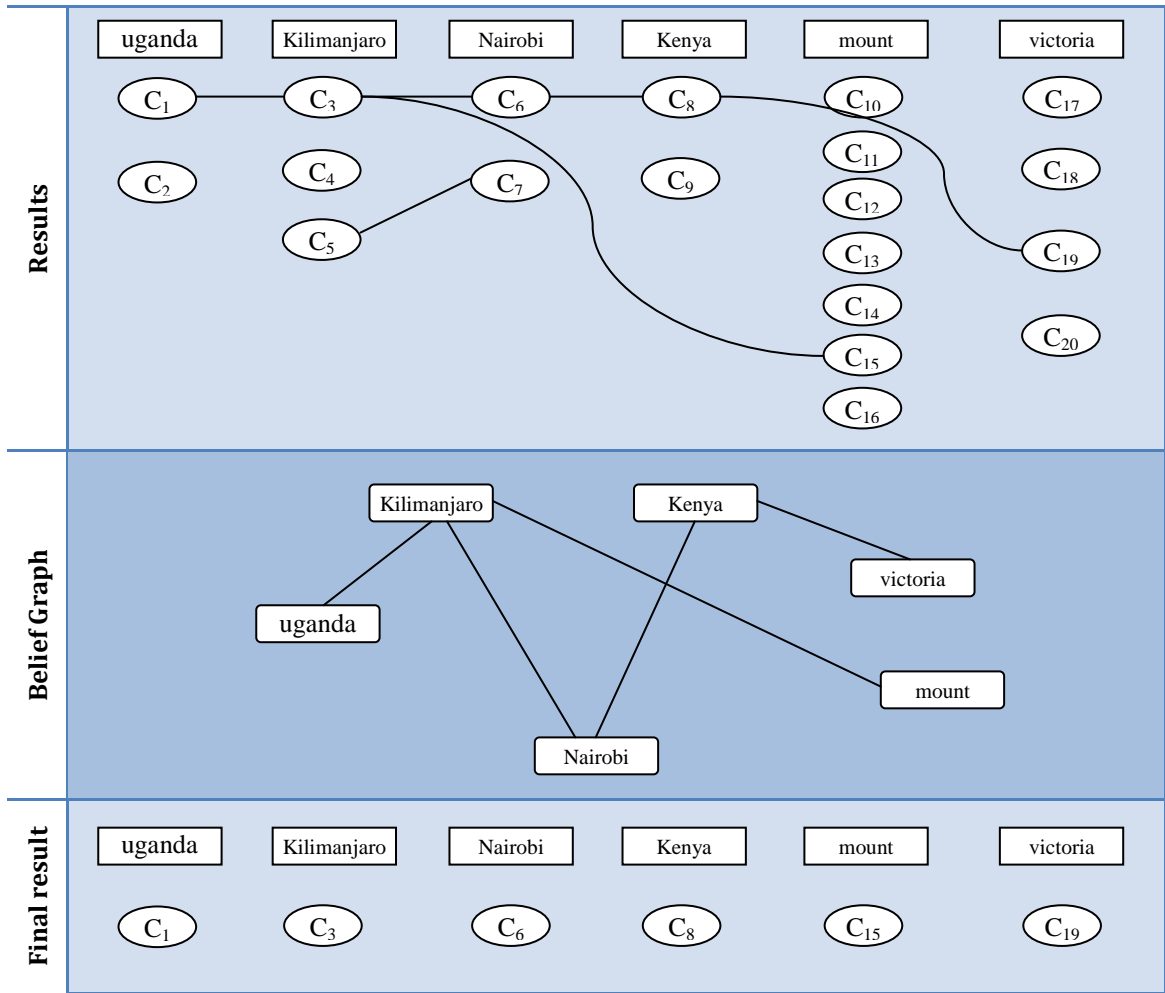


Figure 48 - Disambiguation Example

The results of the disambiguation task are shown in Figure 49 and Table 20.

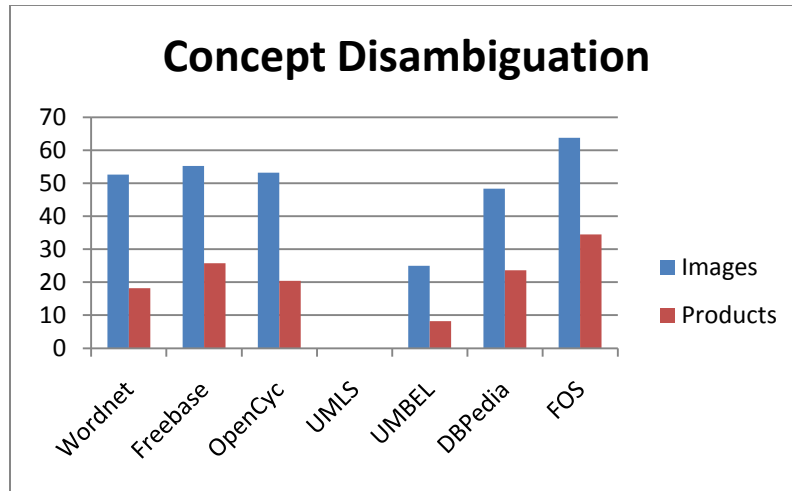


Figure 49 - Disambiguation Results

Disambiguation	Wordnet	Freebase	OpenCyc	UMLS	UMBEL	DBpedia	FOS
Images	52.62	55.23	53.2	0	25	48.34	63.73
Products	18.23	25.76	20.44	0	8.2	23.65	34.5

Table 20 - Disambiguation Increase Percentage

Statistical significant with $p < 0.0001$

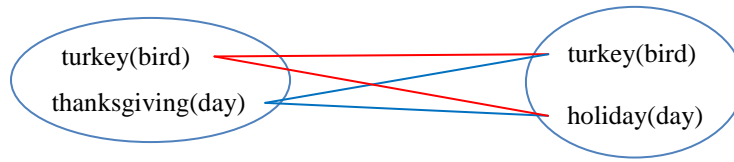
By using the FOS system, we are able to get a 15% increase in the image tags and a 33% increase in the disambiguation of the product tags, even with a conservative process of concept selection. It is important to note that the compounding effect of lack of coverage affects the disambiguation process significantly.

The results do not reflect the upper bound on disambiguation results, since there are multiple techniques that could potentially lead to better results. The goal is to measure the difference between using a baseline and the FOS system.

8.1.1.3 Concept Set Matching

The final task in the matching procedure is to match both sets of concepts after the disambiguation process. Given two sets of concepts, C_1 and C_2 , the goal of this task is to

return a score that describes the semantic similarity of the two sets. In this task we use the *rel* operator to find the similarity between concepts in the different sets.



In order calculate the final score we sum all the distances between the concepts and divide by the total number of concepts. This provides a way to normalize the score. The score is calculated by

$$S = \frac{\sum d(C_i, C_p)}{|i| + |p|}$$

Where S is the score, C_i is the image concept; C_p is the product concept $|i|$ is the number of image concepts and $|p|$ is the number of product concepts. We calculate this score for each product/image pair and re-rank the products according to the generated score.

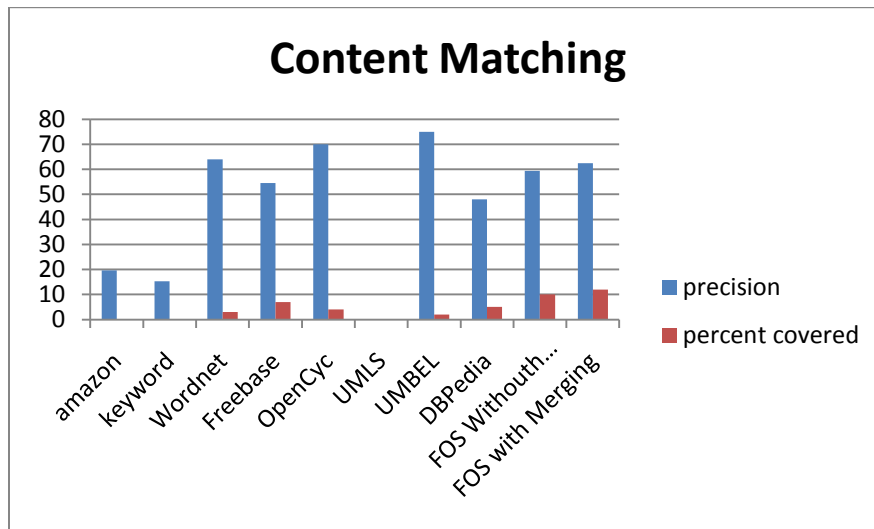


Figure 50 - precision results

Increase	FOS M → NM	FOS M → Freebase
Precision	5.3%	14.5%
Percent coverage	20%	71%

Table 21 - Increase in Matching Accuracy

Content Matching	amazon	keyword	Wordnet	Freebase	OpenCyc	UMLS	UMBEL	DBPedia	FOS No Merging	FOS Merging
Precision	19.6	15.3	64	54.54	70	0	75	48	59.316	62.45
Percent coverage	n/a	n/a	3	7	4	0	2	5	10	12

Table 22 - Results of Content Matching

Statistical significant with $p < 0.001$

Figure 50 shows the results from running the matching algorithm of the Amazon data. We can see a large increase in precision when using the FOS based algorithm when

compared to the Amazon results, the main reason being that Amazon treats each tag as a keyword, using a Boolean query for search. Given the nature of the tags, where we can find different spellings of the same words, amongst other things, in most cases this results in a lack of documents found that match the query terms leading to a back off strategy of using the terms with the highest idf amongst the image keywords. This in turns leads to poor specification, which might help explain Amazon’s poor results.

FOS achieves an increase of 14.5% in precision when compared to Freebase, and more than 100% increase when compared to Amazon and a keyword based baseline. This increase may sound tremendous, but is explicable due to the fact that there are many cases in which the words are semantically related but do not overlap. Some examples are presented below.

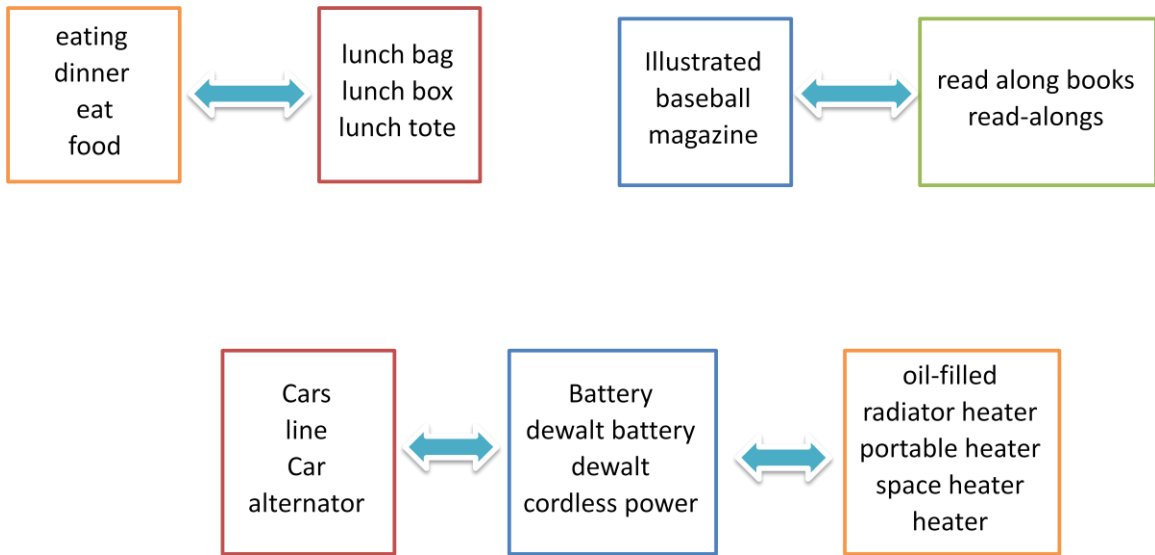


Figure 51 - Examples of semantically related non-overlapping words

As we can observe in figure 48, the boxes on left represent image tags, and the boxes on the right represent product tags. These examples illustrate how we can benefit from using semantic relations to identify relationships between concepts rather than rely on keyword matching.

9 Conclusions and Future Work

9.1 Summary of Contributions

There are many contributions that have been made in the course of our exploration of summarization. Some are summarized in the list below.

- **Ontology Querying Framework**
 - We propose a new approach to ontology querying. We modeled our approach on the concept of federated search and motivated our topic of Federated Ontology Search.
- **Automatic Ontology Creation**
 - We create a new algorithm for automatic ontology creation and demonstrated its uses on the medical domain.
- **Ontology Selection**
 - We proposed a new framework to characterize ontologies from a set of different dimensions that is independent of a particular ontology, allowing for an immediate estimation of the utility of that ontology.
 - We suggest the use of five dimensions to characterize an ontology, namely, scope, depth, cost, noise and connectivity.
 - We proposed the use of pro-active ontology selection to tackle the ontology selection problem
- **Graph Merging**
 - We suggested that there are four basic merging scenarios for ontology merging, depending of the overlapping positions of the source concept.
 - We proposed a procedure that accounts for structural and string merging, focusing on the information need instead of the original source
- **Scoring**
 - We proposed a new algorithm for scoring ontological results that depends on the operators that were used in that query.

- We suggested that we should look at the completeness of the result in order to score its relevancy as a primary measure. We also suggested that both the confidence of the source and the source's confidence on the relations in that result should be taken into account for scoring.
- **Evaluation**
 - We suggested the use of a task based evaluation for evaluating the work performed in this thesis. We evaluated the FOS system on the tasks of type checking, concept coverage, concept disambiguation and content recommendation.
 - We created a data set for ontology performance that consists of 100.000 labeled pairs of matches between tagged products and tagged images.
 - The FOS system achieved a performance increase of up to 100% against a set of proposed set of baselines comprised of using the individual resources.

9.2 Future Work

- **For Automatic Ontology Creation, future research might**
 - Develop algorithms to create ontologies from unstructured data.
 - Explore what relations can adequately be extracted from unstructured text.
 - Extend the proposed algorithm to deal with a larger number of possible relation labels.
- **For Ontology Selection, future research might**
 - Extend the proposed algorithm to adapt the training data to new ontologies automatically, thus reducing the amount of training required.
 - Explore different metrics for ontology selection. How can we accurately predict the best ontologies at a query level?
- **For Ontology Querying, future research might**
 - Extend the current set of operators to account for different phenomena such as exclusion.
 - Study the state explosion problem in the compound operator execution.

- **For result Merging, future research might**
 - Explore machine learning approaches to graph merging applied to ontology merging.
 - Analyze and catalogue the semantic properties of each semantic relation and how can those semantic properties be used to infer semantic distances.
 - Propose new algorithms for ontology merging that might allow for a one-to-many mapping between local graphs.
 - Study automatic transformations for related semantic relations.
 - Explore federated ontology search in a multilingual setting.

9.3 Conclusion

In the last decades the number of available ontologies has grown considerably. Several proprietary and open-domain ontologies have become available. These resources offer the promise of easily-accessible, open-domain ontological information, but the existence of such diverse ontologies raises the issue of information merging and reuse. A comparison of the ontologies reveals both redundant and complementary coverage, but the variety of frameworks and languages used for ontology development makes it a challenge to merge query results from different ontologies. The number of available languages for ontological knowledge engineering such as RDF, OWL, DAML+OIL and CYCL, combined with the existence of independent interfaces aggravates the issue. The lack of a formal way to access and combine the knowledge from different ontologies is an obstacle to more effective re-use and combination of these resources.

In this thesis we proposed an approach to the multi-ontology issue that builds an ontological middleware level for only small fragments of ontologies in an on-demand basis, that is:

- Query multiple ontologies and then merge the query results from multiple knowledge base systems, much like Federated Search in information retrieval (Si and Callan 2005).

- Follow ontological chains and inferences across ontologies, using partial query results from one ontology to query another. This is a more complex version of cross-data-base joins, where the data schemas are sufficiently compatible.

We started by presenting work done in the automatic ontology creation that provides a basis for automatically extending the coverage of the Federated Ontology Search approach. We note that this is but one of the possible methods for ontology acquisition, a field that is currently very active with several methods revealing promise.

We described several issues in ontology selection, result merging and scoring and presented a flexible framework that focuses on parallelizing ontology access and providing a simplified description of the information need for searching ontologies. We proposed a set of operators that allows for a combinatorial construction of complex operators, allowing for rich queries that describe complex information needs, using only simple operators as a basis.

In merging results, we focused on concept normalization, homonym detection and merging with structural differences. Our research suggests that by focusing the results on the need of the user, we are able to bypass some of the more problematic issues that typically crop up in ontology merging. By anchoring the queries in concepts, and dealing with merging after querying the selected ontologies, we were able to avoid a large portion of the ambiguity problems that are typically seen in the field of ontology merging.

We tested our approach in two major tasks one of which consisting of four subtasks. We demonstrated improved results by using the federated approach in type checking, string coverage, concept disambiguation and content matching. As an artifact of this thesis, we create a test set for testing ontologies that will be made available as a by-product of this thesis.

Some of the issues were not within the scope of this thesis, namely compare string comparison algorithms for ontology merging, which certainly have an impact in any

practical application of the work done in this thesis. We have chosen not to dive deep in discussions of ontological formalisms due to the contentious nature of any such formalisms. We believe that ontologies are above all a point of view over a domain, a point of view shared by one or more persons at a given point in time, and as such fundamentally incompatible when considered for a complete merger. Yet, we suggest that by taking pragmatic approach to ontology integration, it is possible to create a set of resources that is of tremendous use to a set of problems that ontologies have always shown promise but often revealed impractical for actual solution.

Within the course of this research, many obstacle were found in the evaluation of a federated ontology set. The lack of standard resources for evaluation hides the fundamental problem with ontology evaluation, the inherent subjective nature of all ontological resources. The authors have worked hard to provide a set of tools that allows other researched to continue to explorer further ways to integrate ontologies and make no claims to the universality of this work. We believe that in the end, the most important part of the work done for this thesis was to define the parameters for the set of deep questions that are yet unanswered. Question such as “what makes a good ontology?”, “what the breadth of possible relations and how are they characterized across ontologies?” “what characterizes the relations between concepts in the same domain?” are some examples of questions that have tremendous impact in the way we will be able process and use ontological knowledge. These questions were not the scope of this thesis, rather we focused on examining the conditions upon which a set of disparate resources can be used effectively to solve real world problems, and what practical concerns arise from such problems. We hope we have succeeded in presenting some interesting new directions for ontology integration and we look forward to contribute further to the field of Federated Ontology Search.

10 Bibliography

- Agirre, E. and P. Edmonds "Word sense disambiguation: Algorithms and applications." Computational Linguistics **33**(2).
- Alani, H. and C. Brewster (2005). "Ontology ranking based on the analysis of concept structures." Proceedings of the 3rd international conference on Knowledge capture: 51-58.
- Avrahami, T. T., L. Yau, et al. (2006). "The FedLemur Project: Federated Search in the Real World." JOURNAL-AMERICAN SOCIETY FOR INFORMATION SCIENCE AND TECHNOLOGY **57**(3): 347.
- Baader, F. and W. Nutt (2003). "Basic description logics."
- Baker, C. F., C. J. Fillmore, et al. (1998). "The Berkeley FrameNet project." Proceedings of COLING-ACL **98**.
- Barwise, J. and J. Seligman (1997). Information flow: the logic of distributed systems, Cambridge University Press.
- Bemers-Lee, T., J. Hendler, et al. (2001). "The Semantic Web." Scientific American **284**(5): 34-43.
- Bhole, A., B. Fortuna, et al. (2007). "Mining Wikipedia and relating named entities over time." Bohanec, M., Gams, M., Rajkovic, V., Urbancic, T., Bernik, M., Mladenic, D., Grobelnik, M., Hericko, M., Kordeš, U., Markic, O. Proceedings of the 10 th International Multiconference on Information Societz IS **8**: 12.
- Blake, C. and W. Pratt (2002). "Automatically Identifying Candidate Treatments from Existing Medical Literature." AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases.
- Boag, S., D. Chamberlin, et al. (2002). "XQuery 1.0: An XML Query Language." W3C Working Draft **15**.
- Bodenreider, O. (2004). "The Unified Medical Language System (UMLS): integrating biomedical terminology." Nucleic Acids Research **32**(1): D267-D270.
- Bollacker, K., C. Evans, et al. (2008). Freebase: a collaboratively created graph database for structuring human knowledge, ACM New York, NY, USA.
- Brank, J., M. Grobelnik, et al. (2005). A survey of ontology evaluation techniques. Conference on Data Mining and Data Warehouses (SiKDD 2005). Ljubljana, Slovenia.
- Callan, J. (2000). Distributed information retrieval. Advances in information retrieval, Kluwer Academic Publishers.
- Callan, J. P., W. B. Croft, et al. (1992). The INQUERY retrieval system, Valencia, Spain.
- Chalupsky, H. (2000). "OntoMorph: A Translation System for Symbolic Knowledge." Principles of Knowledge Representation and Reasoning **185**.
- Converse, T., R. M. Kaplan, et al. "Powerset's Natural Language Wikipedia Search Engine."
- Denoyer, L. and P. Gallinari (2006). "The Wikipedia XML corpus." ACM SIGIR Forum **40**(1): 64-69.

- Ding, L., T. Finin, et al. (2004). "Swoogle: a search and metadata engine for the semantic web." Proceedings of the Thirteenth ACM conference on Information and knowledge management: 652-659.
- Doan, A., J. Madhavan, et al. (2004). "Ontology matching: A machine learning approach." Handbook on Ontologies in Information Systems: 397-416.
- Donmez, P. and J. G. Carbonell (2008). "Proactive learning: cost-sensitive active learning with multiple imperfect oracles."
- Espinoza, M., R. Trillo, et al. "Discovering and Merging Keyword Senses using Ontology Matching." Ontology Matching.
- Fahlman, S. E. (2005). Scone user's manual.
- Fredman, M. L. and R. E. Tarjan (1987). "Fibonacci heaps and their uses in improved network optimization algorithms." Journal of the ACM (JACM) **34**(3): 596-615.
- Freebase. (2007). "Metaweb Query Language(MQL) Reference Guide."
- Fryer, D. (2004). "Federated search engines." Online(Weston, CT) **28**(2): 16-19.
- Ganter, B. and R. Wille (1997). Formal Concept Analysis: Mathematical Foundations, Springer-Verlag New York, Inc. Secaucus, NJ, USA.
- Genesereth, M. R. and R. E. Fikes (1992). Knowledge Interchange Format, Version 3.0 Reference Manual, Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.
- Gómez-Pérez, A. and O. Corcho (2002). "Ontology Languages for the Semantic Web." IEEE INTELLIGENT SYSTEMS: 54-60.
- Gravano, L., P. G. Ipeirotis, et al. (2001). "Using q-grams in a DBMS for Approximate String Processing." IEEE Data Engineering Bulletin **24**(4): 28-34.
- Grosso, W. E., H. Eriksson, et al. (1999). "Knowledge Modeling at the Millennium (The Design and Evolution of Protege-2000)." Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW99): 16-21.
- Guarino, N. (1998). Formal ontology in information systems, IOS Press.
- Halevy, A. Y., Z. G. Ives, et al. (2003). "Piazza: data management infrastructure for semantic web applications." Proceedings of the twelfth international conference on World Wide Web: 556-567.
- Heflin, J., J. A. Hendler, et al. (2003). "SHOE: A blueprint for the semantic web." Spinning the Semantic Web: 29-63.
- Hovy, E. H., M. Fleischman, et al. (2003). The Omega Ontology, prep.
- Izhikevich, E. M. (2007). "Scholarpedia, the free peer-reviewed encyclopedia." from <http://www.scholarpedia.org/>.
- Jiang, J. J. and D. W. Conrath (1997). "Semantic similarity based on corpus statistics and lexical taxonomy." Proceedings of International Conference on Research in Computational Linguistics: 19-33.
- Kalfoglou, Y. and M. Schorlemmer (2002). "Information-flow-based ontology mapping." On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE: 1132-1151.
- Kalfoglou, Y. and M. Schorlemmer (2003). "Ontology mapping: the state of the art." The Knowledge Engineering Review **18**(01): 1-31.
- Kaufman, L. and P. J. Rousseeuw (1990). "Finding groups in data. an introduction to cluster analysis." Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, New York: Wiley, 1990.

- Khoo, C., S. Chan, et al. (2002). "Chapter 4." The Semantics of Relationships: An Interdisciplinary Perspective.
- Kingsbury, P. and M. Palmer (2002). "From Treebank to PropBank." Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002): 1989–1993.
- Klein, M. (2001). "Combining and relating ontologies: an analysis of problems and solutions." Workshop on Ontologies and Information Sharing, IJCAI 1.
- Ko, J., L. Hiyakumoto, et al. (2006). "Exploiting multiple semantic resources for answer selection." Proceedings of of LREC.
- Kozlova, N. (2005). Automatic ontology extraction for document classification, Saarland University.
- Krotzsch, M., D. Vrandečić, et al. (2005). "Wikipedia and the Semantic Web-The Missing Links." Proceedings of Wikimania.
- Lacher, M. and G. Groh (2001). "Facilitating the Exchange of Explicit Knowledge through Ontology Mappings." Proceedings of the 14th International FLAIRS Conference.
- Lenat, D. B. (1995). "CYC: a large-scale investment in knowledge infrastructure." Communications of the ACM 38(11): 33-38.
- Lenat, D. B. and R. V. Guha (1991). "The evolution of CycL, the Cyc representation language." ACM SIGART Bulletin 2(3): 84-87.
- Lih, A. (2003). "Wikipedia as Participatory Journalism: Reliable Sources? Metrics for evaluating collaborative media as a news resource." Nature 2004.
- Lipscomb, C. E. (2000). "Medical Subject Headings (MeSH)." Bull Med Libr Assoc 88(3): 265-266.
- MacGregor, R., H. Chalupsky, et al. (1997). "PowerLoom Manual." ISI, University of South California.
- Maedche, A. and S. Staab (2002). "Measuring similarity between ontologies." Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW): 251–263.
- Maedche, A. D. (2002). Ontology Learning for the Semantic Web, Kluwer Academic Publishers.
- McCarthy, P. (2005). "Search RDF data with SPARQL." from <http://www.ibm.com/developerworks/xml/library/j-sparql/>.
- McGuinness, D. L., R. Fikes, et al. (2000). "The Chimaera Ontology Environment." Proceedings of the 17th National Conference on Artificial Intelligence.
- McGuinness, D. L. and F. van Harmelen (2004). "OWL Web Ontology Language Overview." W3C Recommendation 10: 2004-2003.
- Mena, E., A. Illarramendi, et al. (2000). "OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-Existing Ontologies." Distributed and Parallel Databases 8(2): 223-271.
- Miller, G. A. (1995). "WordNet: A Lexical Database for English." COMMUNICATIONS OF THE ACM 38: 11-39.
- Milne, D., O. Medelyan, et al. (2006). "Mining Domain-Specific Thesauri from Wikipedia: A Case Study." Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence: 442-448.
- Mueller, E. T. (1997). Natural Language Processing with Thoughttreasure, Signiform.

- Murphy, K., Y. Weiss, et al. (1999). Loopy belief propagation for approximate inference: An empirical study.
- Navigli, R., P. Velardi, et al. (2003). "Ontology learning and its application to automated terminology translation." Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications] **18**(1): 22-31.
- Niles, I. and A. Pease (2001). "Towards a standard upper ontology." Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001: 2-9.
- Noy, N. F. (2004). "Semantic integration: a survey of ontology-based approaches." ACM SIGMOD Record **33**(4): 65-70.
- Noy, N. F. and M. A. Musen (1999). "SMART: Automated Support for Ontology Merging and Alignment." Twelfth Workshop on Knowledge Acquisition, Modeling, and Management, Banff, Canada.
- Noy, N. F. and M. A. Musen (2000). "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment." Proceedings of the National Conference on Artificial Intelligence (AAAI).
- Noy, N. F. and M. A. Musen (2002). "PromptDiff: A fixed-point algorithm for comparing ontology versions." 18th National Conference on Artificial Intelligence (AAAI-2002).
- Nyberg, E., T. Mitamura, et al. (2003). "The javelin question-answering system at trec 2003: A multi-strategy approach with dynamic planning." Proceedings of the Twelfth Text REtrieval Conference (TREC2003).
- Patel, C., K. Supekar, et al. (2003). "OntoKhoj: a semantic web portal for ontology searching, ranking and classification." Proceedings of the fifth ACM international workshop on Web information and data management: 58-61.
- Pedro, V., S. Niculescu, et al. (2008). Okinet: Automatic Extraction of a Medical Ontology From Wikipedia.
- Philpot, A., M. Fleischman, et al. (2003). "Semi-Automatic Construction of a General Purpose Ontology." Proceedings of the International Lisp Conference, New York, NY. Invited.
- Prasad, S., Y. Peng, et al. (2002). "Using explicit information to map between two ontologies." Proceedings of the AAMAS 2002 Workshop on Ontologies in Agent Systems (OAS'02): 52-57.
- Quillian, M. R. (1967). "Word concepts: a theory and simulation of some basic semantic capabilities." Behav Sci **12**(5): 410-430.
- Raymond, J. W., E. J. Gardiner, et al. (2002). "RASCAL: Calculation of Graph Similarity using Maximum Common Edge Subgraphs." The Computer Journal **45**(6): 631-644.
- Reed, S. and D. Lenat (2002). "Mapping ontologies into cyc." AAAI 2002 Conference Workshop on Ontologies For The Semantic Web, Edmonton, Canada, July.
- Resnik, P. (1995). "Using information content to evaluate semantic similarity in a taxonomy." Proceedings of the 14th International Joint Conference on Artificial Intelligence **1**: 448-453.
- Ross, S. M. (1976). A first course in probability, Macmillan.
- Sanger, L. (2007). "Citizendium." from www.citizendium.org.

- Schuler, K. K. (2003). VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon, Ph. D. thesis proposal, University of Pennsylvania.
- Serafini, L. and A. Tamilin (2005). "Drago: Distributed reasoning architecture for the semantic web." Proc. of the Second European Semantic Web Conference (ESWC'05) **3532**: 361–376.
- Si, L. and J. Callan (2005). "Modeling search engine effectiveness for federated search." Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval: 83-90.
- Snow, R., D. Jurafsky, et al. (2005). Learning syntactic patterns for automatic hypernym discovery.
- Spackman, K. A., K. E. Campbell, et al. (1997). "SNOMED RT: A reference terminology for health care." Proc AMIA Annu Fall Symp **640**(4): 503-512.
- Staab, S. (2004). Handbook on Ontologies, Springer.
- Stumme, G. and A. Maedche (2001). "FCA-Merge: Bottom-up merging of ontologies." 7th Intl. Conf. on Artificial Intelligence (IJCAI'01): 225–230.
- Völkel, M., M. Krötzsch, et al. (2006). "Semantic Wikipedia." Proceedings of the 15th international conference on World Wide Web: 585-594.
- Voorhees, E. M. (2003). "Overview of the TREC 2003 Question Answering Track." Proceedings of the Twelfth Text REtrieval Conference (TREC 2003).
- Yates, A., M. Cafarella, et al. (2007). "TextRunner: Open Information Extraction on the Web." Proceedings of the NAACL HLT Demonstrations Program.
- Zhang, Y., W. Vasconcelos, et al. (2004). "Ontosearch: An ontology search engine." Proc. 24th SGAI Int. Conf. on Innovative Techniques and Applications of Artificial Intelligence.