

Improving Ontology Matching Towards Achieving Semantic Interoperability

By

Aminu Mustapha Bagiwa
Department Of Mathematics, Faculty Of Science, Ahmadu Bello
University, Zaria

June, 2011

Improving Ontology Matching Towards Achieving Semantic Interoperability

By

Aminu Mustapha Bagiwa(B.Sc Computer Science) UDUS
M.Sc/Scien/00169/2008-09

A Thesis Report Submitted To The Postgraduate School, Ahmadu Bello
University In Partial Fulfillment For The Award Of M.Sc Degree In
Computer Science.

Department Of Mathematics

DECLARATION

I declare that the work in this project report entitled Improving Ontology Matching Towards Semantic Interoperability has been performed by me in the Department of Mathematics under the supervision of Professor S.B Junaidu and Dr. M.I. Bukhari. The information derived from the literature has been duly acknowledged in the text and list of references provided. No part of this project report was previously presented for another degree or diploma at any university.

Mustapha Aminu Bagiwa

Name of student

Signature

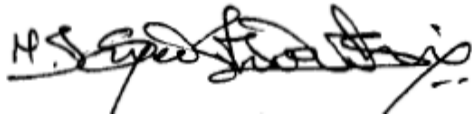
Date

CERTIFICATION

This project report entitled IMPROVING ONTOLOGY MATCHING TOWRDS SEMANTIC INTEROPERABILITY by AMINU, Mustapha Bagiwa meets the regulation governing the award of the degree of M.Sc Computer Science of Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

Professor S.B. Junaidu
Major Supervisor

Date



Dr. M.I. Bukhari
Minor Supervisor

Date

Dr. A. A. Tijjani
H.O.D Mathematics

Date

Professor A. A. Joshua
Dean, Postgraduate School

Date

DEDICATION

This thesis is dedicated to my beloved parents in the name of Alhaji Aminu Idris Bagiwa and Hajia Hadiza Aminu Bagiwa for not only their financial support but for the time and resources they spent in training and bringing me up to be a person of value, importance to the society and life in general. Finally this thesis is also dedicated to my entire family and a special person whom i cannot forget to mention in the name of Halima Kabir.

ACKNOWLEDGEMENT

In the name of Allah, the most merciful. All gratitude is due to him for giving me this wonderful opportunity to live up to this time and enable me obtain this honor. He is the guider and the protector. I acknowledge first the effort of my humble supervisors Professor S.B. Junaidu(Major) and Dr. M.I. Bukhari(Minor) for sacrificing their efforts and time to see me through the completion of this thesis.

I also acknowledge the effort of the entire lecturers of the department of mathematics for their academic and intellectual support towards the success of this thesis. A special acknowledgment goes to my parents, Mukhtar Aminu Bagiwa, Idris Aminu Bagiwa, Abdulhakeem Aminu Bagiwa, Aisha Aminu Bagiwa, Ramatu Aminu Bagiwa, Halima Kabir and my Uncle Alhaji Sani Idris Bagiwa for their financial and moral support for easy running of my entire study programme. I cannot forget mentioning some of my colleagues who contributed one way or the other towards the completion of this thesis, people like Halima Dan-Abdul, Barroon Isma'eel Ahmad, Salihu Idi Dishing and my entire class members.

A special acknowledgment also goes to A.B.U Mc Arthur foundation for their financial support towards the successful completion of this thesis.

Finally, I thank you all. May almighty Allah be with you in all your endeavors.

ABSTRACT

This work proposes an algorithm for concept matching, applied in the ontology mapping domain. The basic idea is to seek the effective semantics embedded in the concept name by analyzing the contexts in which it appears. Through simple interaction with the known lexicon WordNet, the right meaning associated with a concept is unequivocally elicited by exploring their local semantic contexts. This approach reveals interesting results for the word sense disambiguation, when polysemy problems require a semantic interpretation. The algorithm, though takes a longer time but yet produce a better matching because the concepts in the ontology trees are populated with much semantic information at the end of the first and second step of the matching process.

Table of Contents

DECLARATION	iii
CERTIFICATION.....	iv
DEDICATION	v
ACKNOWLEDGEMENT.....	vi
ABSTRACT.....	vii
Table of Contents.....	viii
LIST OF TABLES.....	xi
LIST OF FIGURES.....	xii
LIST OF APPENDICES	xiii
ABBREVIATIONS, DEFINITIONS, GLOSSARIES AND SYMBOLS	xiv
CHAPTER ONE	1
GENERAL INTRODUCTION.....	1
1.1 Background of the study.....	1
1.2 Research motivations and goals	2
Motivating Example	3
1.3 Research questions	9
1.4 Research objectives	10
1.5 Methodology.....	11
1.6 Contribution to Knowledge.....	11
CHAPTER TWO	12
LITERATURE REVIEW.....	12
2.1 Designing and classifying ontologies.....	12
2.1.1 Why develop ontologies	13
2.1.2 Steps in developing ontologies	13
2.1.3 Problems and solutions to ontology development	14
2.1.4 Ontologies in the semantic web language.....	14
Example	15
2.1.5 Owlvisualizer (Owlviz).....	15
2.2 Ontology interoperability.....	15
2.3 Ontology matching towards semantic interoperability.....	16
2.3.1 What is ontology matching?	18
2.3.2 Why is ontology matching needed/interesting?	19

Language or meta-model level.....	19
Ontology or model level	19
2.3.3 What is semantic interoperability, why is it needed?.....	20
2.4 Review of ontology matching techniques.....	20
2.4.1 Glue.....	20
2.4.2 Coma++	21
2.4.3 Automatch.....	21
2.4.4. Duma.....	22
2.4.5 Scalable knowledge composition.....	22
2.5 Comparisons of ontology matching techniques	22
2.6 Limitations of ontology matching techniques	23
CHAPTER THREE	25
SYSTEM DEVELOPMENT	25
3.1 System Requirement.....	25
3.1.1 Protégé Ontology Software Tool.....	26
3.1.2 S-match	31
3.2 Introduction to graphs	32
3.2.1 Textual representation of graphs	35
3.3 Pattern in graphs.....	36
3.3.1 Pattern specification and detection.....	36
3.3.2 Pattern visualization and navigation.....	37
CHAPTER FOUR	38
SYSTEM IMPLEMENTATION	38
4.1 WordNet Overview	38
4.2 The semantic matching algorithm	39
4.3 The tree matching algorithm: Step I Computing concepts at labels.....	42
4.4 Dealing with ambiguity using concept sense discrimination algorithm	44
4.5 The computation of the C_L matrix.....	46
4.6 The computation of the C_N matrix	48
4.7 Discussion.....	50
4.8 An Architecture of our enhanced S-match implementation	52
4.9 Conclusion.....	54
CHAPTER FIVE.....	55
CONCLUSION AND RECOMMENDATION	55

5.1 Conclusion and future work.....	55
5.2 Recommendation.....	55
REFERENCE.....	56
APPENDIX ONE	58

LIST OF TABLES

Table 1: Semantic Relation Between Concepts Of Tree A And Tree B	48
Table 2: S-Match and Enhanced S-match Comparison	51

LIST OF FIGURES

Figure 1:Computer science Department ontology for Niger.....	4
Figure 2: Computer science Department ontology for Nigeria.....	5
Figure 3:Architectural operation of the semantic web.....	6
Figure 4: A cycle flow of semantic information across ontologies	18
Figure 5: An image ontology domain	24
Figure 6:An image ontology domain	24
Figure 7: Class hierarchy for our course ontology using protégé tool.....	27
Figure 8: Inferred model for our course ontology using protégé tool.....	28
Figure 9: Class hierarchy for two different course ontologies.....	29
Figure 10: Class hierarchy for two different university ontologies	30
Figure 11: Class hierarchy for two different picture ontologies	31
Figure 12: Class hierarchy of course ontology with textual annotations	34
Figure 13: Tree representation of course ontology with textual annotations.....	35
Figure 14: Architecture of the enhanced S-Match Platform	53

LIST OF APPENDICES

Appendix 1.....	59
-----------------	----

ABBREVIATIONS, DEFINITIONS, GLOSSARIES AND SYMBOLS

Meronym - This is the semantic relation that holds between a part and the whole class

Hyponym – This is the semantic relation of being subordinate or belonging to a lower rank or class

S-match – Semantic Match

WN - WordNet

Synset – Synonym Set

P.O – Post Office

WWW – world Wide Web

EQ – Equivalence

DJ – Disjointness

MG – More General

LG- Less General

CHAPTER ONE

GENERAL INTRODUCTION

This chapter discusses the introductory part of the thesis which includes the background of the study, research motivations and goals, the research questions for which the thesis should provide answers to, the methodology that is used to answer those questions and finally the summary of the thesis contribution to knowledge.

1.1 Background of the study

The world wide web is the greatest repository of information ever assembled by man. It contains documents and multimedia resources concerning almost every imaginable subject, and all of these data are instantaneously available to anyone with an Internet connection. The web's success is largely due to its decentralized design: web pages are hosted by numerous computer, where each document can point to other documents, either on the same or different computers. As a result, individuals all over the world can provide content on the web, allowing it to grow exponentially as more and more people learn how to use it. However, the web's size has also become its limitation. Due to the sheer volume of available information, it is becoming increasingly difficult to locate useful information. Although directories (such as Yahoo!) and search engines (such as Google and Alta Vista) can provide some assistance, they are far from perfect. For many users, locating the right document is still like trying to find a needle in a sea. Ontologies represent a conceivable solution for data representation as well as the knowledge sharing, aimed at the integration of the web content in a unique and coherent view. Nevertheless, due to the decentralized nature of the web, a number of ontologies have been defined and

disseminated on the Internet; often they describe overlapped application domains; sometimes are specialized for specific domain.

Noy *et.al* (2010) states that it is evident the exigency to find some semantic correspondence among concepts which refer to different ontologies in order to get a semantic reconciliation, aimed at establishing interoperability between semantic web applications and a more homogeneous integration of information.

The main obstacle is the fact that the web was not designed to be processed by machines. Although web pages include special information that tells a computer how to display a particular piece of text or where to go when a link is clicked, they do not provide any information that helps the machine to determine what the text means. Thus, to process a web page intelligently, a computer must understand text, but natural language understanding is known to be an extremely difficult and unsolved problem. In order for machines to be able to integrate information that commits to heterogeneous ontologies, there need to be primitives that allow ontologies to map terms to their equivalents in other ontologies.

1.2 Research motivations and goals

The goal of the semantic web is to take advantage of formalized knowledge (in languages like RDF) at the scale of the world wide web. In particular, it is based on ontologies which define concepts used for representing knowledge on the web, e.g., for annotating a picture, specifying a web service interface or expressing the relation between two persons.

Some researchers and web developers have proposed that we augment the web with languages that make the meaning of web pages explicit. Tim Berners-Lee, inventor of the Web, has coined the term semantic web to describe this approach. Wang *et.al*(2008) provide the following definition: The semantic web is not a separate web but an extension of the current one, in which

information is given a well-defined meaning, better enabling computers and people to work in cooperation.

The following are some of the problems the semantic web is meant to solve:

1. allow users to organize and browse the web in ways that are more suitable to the problems they have at hand.
2. impose a conceptual filter to a set of web pages, and display their relationships based on such a filter.
3. allow visualization of complex content.

Motivating Example

Suppose you want to find out more about someone you met at a conference. You know that his last name is Aminu, and that he teaches Computer Science at a nearby university, but you do not know which one. You also know that he just moved to Nigeria from Niger, where he had been an associate professor at his alma mater.

On the world wide web of today you will have trouble finding this person. The above information is not contained within a single web page, thus making keyword search ineffective. On the semantic web, however, you should be able to quickly find the answers. A marked-up directory service makes it easy for your personal software to find nearby computer science departments.

Here the data is organized into a taxonomy as in Figure 1 and Figure 2 that includes staff, Academic, and Non Academic. Associate professors have attributes such as name, degree, and degree-granting institution. Such marked-up data makes it easy for your software to find a professor with the last name Aminu. Then by examining the attribute granting

institution, the software quickly finds the alma mater CS department in Niger. Here, software learns that the data has been marked up using an ontology specific to Nigerian universities, and that there are many entities named Aminu. However, knowing that `asistant professor` is equivalent to `senior lecturer`, machines can select the right sub tree in the departmental taxonomy, and zoom in on the old homepage of your conference acquaintance.

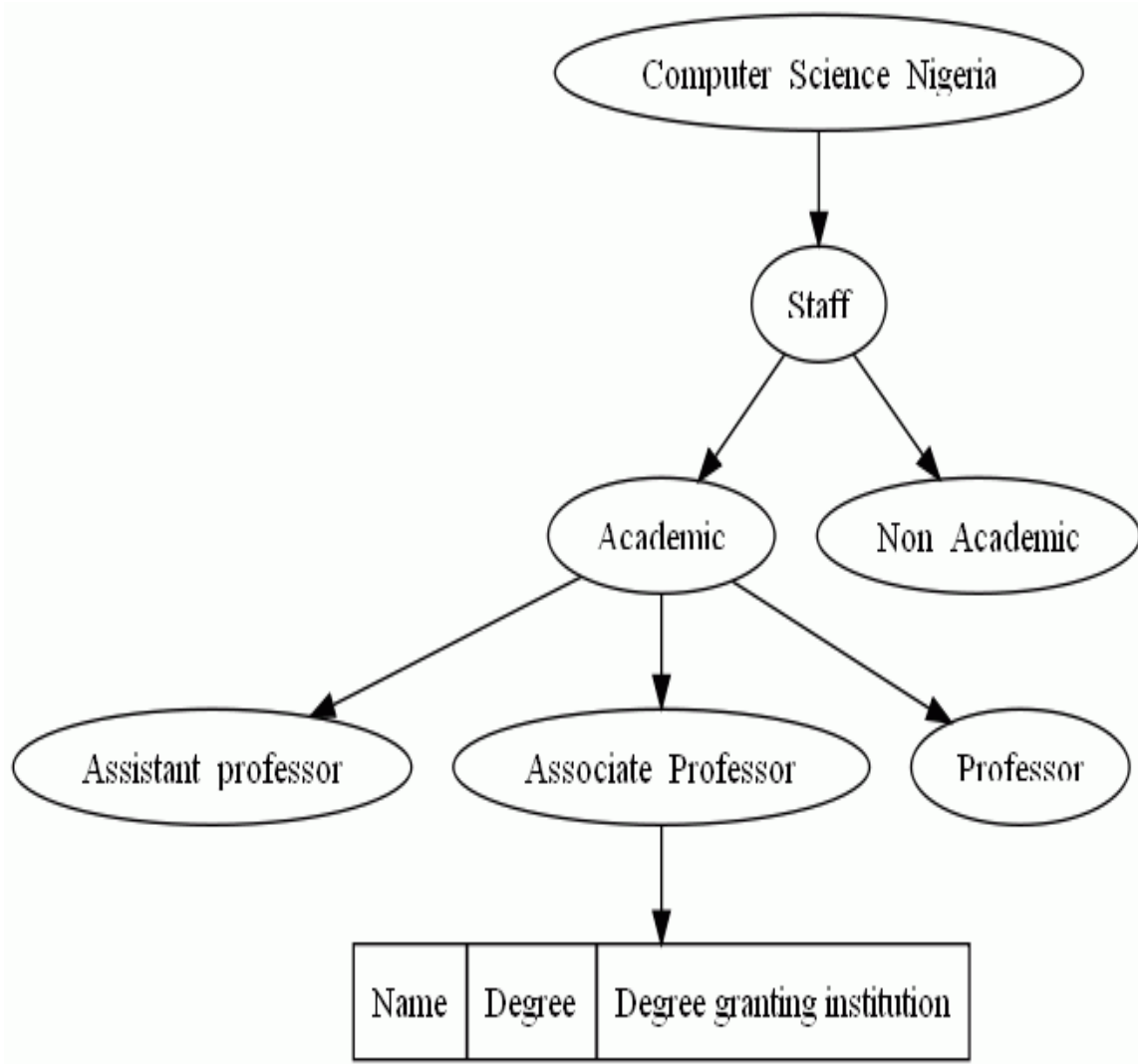


Figure 1:Computer science Department ontology for Niger

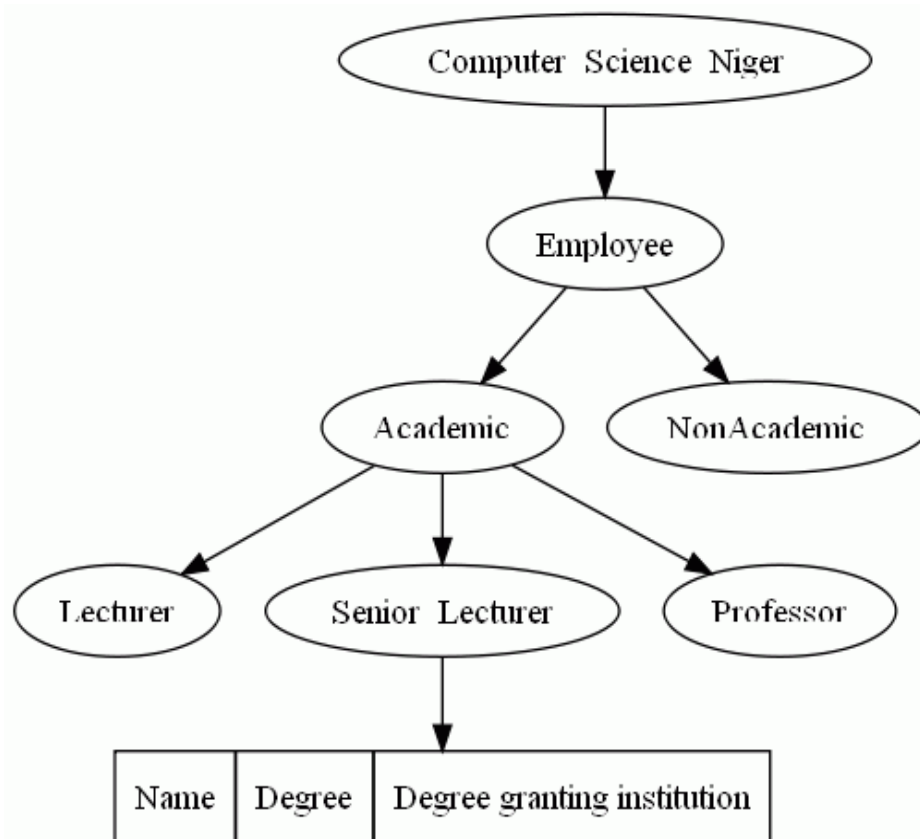


Figure 2: Computer science Department ontology for Nigeria

The semantic web depends on the ability to associate formal meaning with content. The field of knowledge representation provides a good starting point for the design of a semantic web language because it offers insight into the design and use of languages that attempt to formalize meaning.

The idea here should be clear that our software agents are not a replacement of humans in the use of the semantic web because they can not on their own make certain decisions but the only role they play is to harmonize information on the web so that users can make easy decisions. This brings the fact that the semantic web does not only intend to represent knowledge and formal semantics but also performs some reasoning based on certain artificial intelligence

concept. Figure3 shows the flow of information between users, personal agents, intelligent infrastructure and the web document we create.

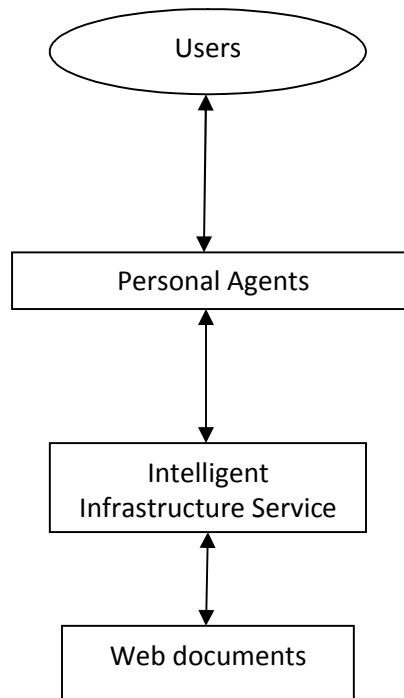


Figure 3: Architectural operation of the semantic web

One of the driving factors in the proliferation of the web is the freedom from a centralized authority. However, since the web is the product of many individuals, the lack of central control presents many challenges for reasoning with its information. First, different communities will use different vocabularies, resulting in problems of synonym (when two different words have the same meaning) and polysem (when the same word is used with different meanings). Second, the lack of editorial review or quality control means that each page’s reliability must be questioned. An intelligent web agent simply cannot assume that all of the information it gathers is correct and consistent. There are quite a number of well-known “web hoaxes” where information was published on the web with the intent to amuse or mislead. Furthermore, since there can be no global enforcement of integrity constraints on the web, information from different sources may be in conflict. Some of these conflicts may be due to philosophical disagreement; different

political groups, religious groups, or nationalities which may have fundamental differences in opinion that will never be resolved.

In order for information from different sources such as websites to be integrated, there needs to be a shared understanding of the relevant domain. Knowledge representation formalisms provide structures for organizing this knowledge, but provide no mechanisms for sharing it. Ontologies provide a common vocabulary to support the sharing and reuse of knowledge, As discussed by Guarino and Giaretta(1995), the meaning of the term ontology is often vague. It was first used to describe the philosophical study of the nature and organization of reality. In AI, the most cited definition is due to Gruber *et al.*(2009): “An ontology is an explicit specification of a conceptualization.”.

The semantic web thus offers a compelling vision, but it also raises many difficult challenges. Researchers have been actively working on these challenges, focusing on fleshing out the basic architecture, developing expressive and efficient ontology languages, building techniques for efficient marking up of data, and learning ontologies. A key challenge in building the semantic web, one that has received relatively little attention, is finding semantic mappings among the ontologies. Given the de-centralized nature of the development of the semantic web, there will be an explosion in the number of ontologies. Many of these ontologies will describe similar domains, but using different terminologies and others will have overlapping domains. To integrate data from disparate ontologies, we must know the semantic correspondences between their elements.

In the last decade, the Semantic web initiative has grown from the personal vision of a small group of individuals to a large research field with several international conferences, multiple journals, and a wide range of interesting research projects all over the world. The Semantic web

can be conceived as a collection of ontologies. Ontologies are generally used to specify and communicate domain knowledge in a generic way.

This is the key technology for realizing the semantic web . This approach forces taxonomic hierarchies, where it describes what things are and what they are used for. The current ontology technology is matured enough to provide means for development, management and reasoning within the single ontology of a particular organization. Different ontologies may be modeled for the same concepts in different ways. Although shared ontologies and ontology extension allow a certain degree of interoperability between different organizations and domains, there are often cases where there are multiple ways to model the same information. This may be due to differences in the perspectives of different organizations, different professions, different nationalities, etc.

In order for machines to be able to integrate information that commits to heterogeneous ontologies, there need to be primitives that allow ontologies to map terms to their equivalents in other ontologies. Ontology matching is one of the core tasks for ontology interoperability. It is aimed to find semantic relationships between entities (i.e. concept, attribute, and relation) of two ontologies.

Many different matching solutions have been proposed so far, some of which include:

Similarity Flooding. Melnik *et al.*(2002) approach utilizes a hybrid matching algorithm based on the ideas of similarity propagation. Schemas are presented as directed labeled graphs; grounding on the OIM specification by Meta *et al.*(1999) the algorithm manipulates them in an iterative fixed-point computation to produce an alignment between the nodes of the input graphs.

COMA.(Combination of Matching algorithms) COMA by Do *et al.*(2002) is a composite schema matching tool. It provides an extensible library of matching algorithms; a framework for combining obtained results, and a platform for the evaluation of the effectiveness of the different matchers.

S-Match. S-Match by Guindiglia and Shvaiko(2003) is a schema-based matching system. It takes two graph like structures (e.g., XML schemas or ontologies) and returns semantic relations (e.g., equivalence, subsumption) between the nodes of the graphs that correspond semantically to each other. The relations are determined by analyzing the meaning (concepts, not labels) which is codified in the elements and the structures of schemas/ontologies.

The goals of this thesis work is to extend the S-match ontology matching algorithms with automatic learning techniques in order to make data more sharable and to facilitate the communication between ontology applications using an enhanced matching technique.

1.3 Research questions

The study is set up to answer the following questions:

1. What automatic learning techniques are most appropriate for incorporation into ontology matching algorithms?

Ontology matching is a major problem when it comes to ontology integration. Many ontology matching techniques have been developed based on certain reasoning algorithms.

The interesting point is that almost all these algorithms use some concept of similarity measures to calculate a joint probability distribution that exist between concepts. Uniform textual representation can be used to enhance ontology matching. This type of representation can be incorporated into graph based approach of ontology matching.

2. Which ontology matching algorithm admits learning techniques with a minimal overhead?

Graph based ontology matching algorithms classify input ontologies as labeled graphs containing nodes and edges. The nodes contain labels associated to them that provide additional information to the classes they represent in that concept. This is a perfect algorithm for adding machine learning technique as edges can be associated with certain labels representing some particular patterns

3. How can textual annotations be added to ontology matching algorithms to make data between ontologies more sharable?

The idea is to be able to add some textual labels to nodes of graph(representing ontologies) so that software systems are able to deduce some additional meaning from the input ontologies. Our approach uses simple logical comparisons (rather than joint probability calculations used in many other researches) to make inference.

1.4 Research objectives

The main objective of this research include:

1. Defining an architectural framework for adding textual annotations to hierarchical trees representing ontologies of different domain.
2. Enhancing the s-match algorithm to understand those textual annotations.
3. Implementation of the enhanced s-match.
4. Evaluation of the two implementations.

1.5 Methodology

The following are the steps that are set out to answer the above research questions:

1. Review literature to understand why it is necessary to match ontologies.
2. Conduct extensive literature review on ontology matching techniques.
3. Create and read simple ontologies using protégé which is one of the most widely used ontology editing tool.
4. Determine unsatisfiable classes and providing explanation for unsatisfiability in terms of matching ontologies.
5. Develop an algorithm to enhance ontology matching based on machine leaning approach.

1.6 Contribution to Knowledge

The thesis contributes to the accuracy and method of ontology interoperability through matching concepts in some domain of interest. Furthermore it demonstrates the fact that human intervention can be minimized interms of ontology matching by adding some automatic learning techniques to ontology matching algorithms.

CHAPTER TWO

LITERATURE REVIEW

This chapter discusses the importance of designing and classifying ontologies, the steps that involves ontology design, the possible problems that can be encountered in the process of ontology development. The usage of ontology as the semantic web language is also discussed. The chapter also takes a look at interoperability among ontologies which result in the ontology matching problem. The chapter also discusses why it is necessary to match ontologies, earlier techniques of ontology matching techniques were discussed and compared to our own approach. Finally the limitations of the techniques are discussed.

2.1 Designing and classifying ontologies

The development and design of ontologies provides an explicit formal specifications of terms in some domain of interest and specify relations that exist between them. Ontologies are becoming important and useful on the web. The WWW Consortium (W3C) has developed the Resource Description Framework, a language for encoding knowledge on web pages to make it understandable to machines searching for information on the web. Many educational disciplines now design and classify standardized ontologies that domain experts can use to share and add critical information in their fields. Medicine, for example, has produced large, standardized, structured vocabularies such as SNOMED Price and Spackman (2000) and the semantic network of the Unified Medical Language System.

2.1.1 Why develop ontologies

An ontology defines a common vocabulary to share information in a domain of discourse. It includes machine-interpretable definitions of basic concepts in the domain and relations that exist among them.

Why would someone want to develop an ontology? Some of the reasons are:

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge

2.1.2 Steps in developing ontologies

Ontology development requires defining a set of data and specifying a unified structure for other programs to use. Problem-solving methods, domain-independent applications, and software agents use ontologies and knowledge bases built from ontologies as input data for various applications.

It is quite tough to specify all the steps in ontology development, but the following are some major steps that any developer of ontology should adhere to:

- Identify the domain for which ontologies should be built
- Identify and design classes in the ontology
- Arranging the classes in a taxonomic (subclass–super class) hierarchy,
- Defining slots and describing allowed values for those slots,
- Filling in the values for slots and for instances

2.1.3 Problems and solutions to ontology development

The major problems surrounding ontology development include the following:

- There is no one correct way to model a domain, there are always viable alternatives. The best solution almost always depends on the application that you have in mind and the extensions that you anticipate.
- Ontology development is necessarily an iterative process.
- Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain.

2.1.4 Ontologies in the semantic web language

There is good history behind ontologies in the field of philosophy and computer science. Researches are still on going to find a means of including knowledge representations and intelligence to the world wide web. The major objective is to find some semantic correspondences between vast amount of information stored on the web. These can be achieved using quite different ontology languages, most of which are based either on HTML(Simple HTML Ontology Extension), XML(Ontology Inference Language), or frame based Knowledge representation and acquisition languages. Ontologies are some explicit specification of conceptualization due to Gruber(1993). Ontologies are key issues in the area of semantic web, they can always be used in application such as information retrieval, web services, web management. Ontologies could also be used to tackle the problem of interoperability between software systems.

Example

An ontology describing universities can have axioms that “a taught by” property is only allowed between a class called course and another class called academic staff.

2.1.5 Owlvisualizer (Owlviz)

Owl is a program that is designed to be used by protégé software which is an ontology editing tool. It is a graphical user interface that allows a class hierarchy to be viewed and navigated. It also allows comparisons between different class hierarchies. In protégé 4.0.2, ontology is implemented as a view which allows the display of subsumption graphs. The following are some of the features involve in Owl:

- Display of asserted and inferred hierarchy.
- Exporting classes as image.
- Viewing imported hierarchy.

2.2 Ontology interoperability

The primary aim of ontology development is to allow applications to determine and agree on a unified means of communication. This means that these ontologies have to be shared between certain application domains so that the sharing can take place not only at the syntactic level but also at the semantic level. Ontology interoperability can be seen as a way of transferring information between multiple applications, which may or may not rely on different conceptualization of domain of interest. Ontology building today is a fragmented practice. The situation, in part, is as a result of the proliferation of logic languages and information models that have combined to yield even more ontology forms and editing environments. These tools and methodologies, along with the ontologies built with them, generally exist without proven

interoperability. This is one of the challenges facing the practice along with establishing methods to integrate ontology components with enterprise information systems and standards.

Ontologies are for sharing. They are intended to serve as consensual rallying points to exchange and interpret information. Clearly, the wider the range of applications and other ontologies that can use an ontology, the greater its utility and the mutual utility of the interrelating ontologies. This requires formal compatibility on syntactic levels as well as semantic levels. One consideration in the enterprise realm, for example, is the ability of a domain ontology to accommodate specialized XML language and controlled vocabularies being adopted as standards in various industries. None of the current ontology editors address this capability fully, however vendors like Modulant and Unicorn are moving in this direction.

Interoperability, instead, is being addressed simply through an editor's ability to import and export ontologies in different language serializations. Some tools like Stanford Knowledge Systems Lab's Ontolingua offers a wide range of translations, while most are limited. Importing or exporting ontologies in the newer languages like DAML+OIL and OWL usually means that the translation is only partial and expressiveness is lost. A few editors like Web ODE also offer heterogeneous ontology merging capabilities.

2.3 Ontology matching towards semantic interoperability

The semantic web community agrees on the fact that a single, universal ontology can not be built. Because of the large variety of information sources on the web, documents on it will inevitably result from many different ontologies. We foresee that in the future most information systems will use ontologies. We must expect an explosion in the number of ontologies, even when considering similar domains. For these reasons, a key challenge in building the semantic web is enabling the interoperability among different ontologies. Ontologies can interoperate only

if correspondences between their elements have been identified and established. Today, if two ontologies need to interoperate, the mapping is mainly achieved by hand. But this task is tedious, error-prone, and time consuming. According to the above scenario, the manual solution of the ontology interoperability problem could be a bottleneck in building a network of cooperating information management systems. Hence, the introduction of new methodologies and user-friendly tools that support the knowledge engineer in discovering semantic correspondences is crucial to the success of the semantic web.

Several solutions have been proposed in literature to address the ontology interoperability problem. Different methods tackle different aspects of the problem. Some of the existing approaches are aimed at enabling interoperability at the language level. It means they try to map the formalisms used to represent the ontologies, in order to homogenize the descriptions and then to compare the elements belonging to the different ontologies, some provide transformation rules between languages constructs. Several approaches are interactive and suggest to the user possible alignments and mapping. Among them, we can find linguistic based approaches and approaches based on the analysis and evaluation of the structural and model similarity.

Instead of aligning two ontologies “directly” through their signatures, our result specifies the mapping of two ontologies O_1 and O_2 by means of a pair of ontology mappings from an intermediate source ontology O_3

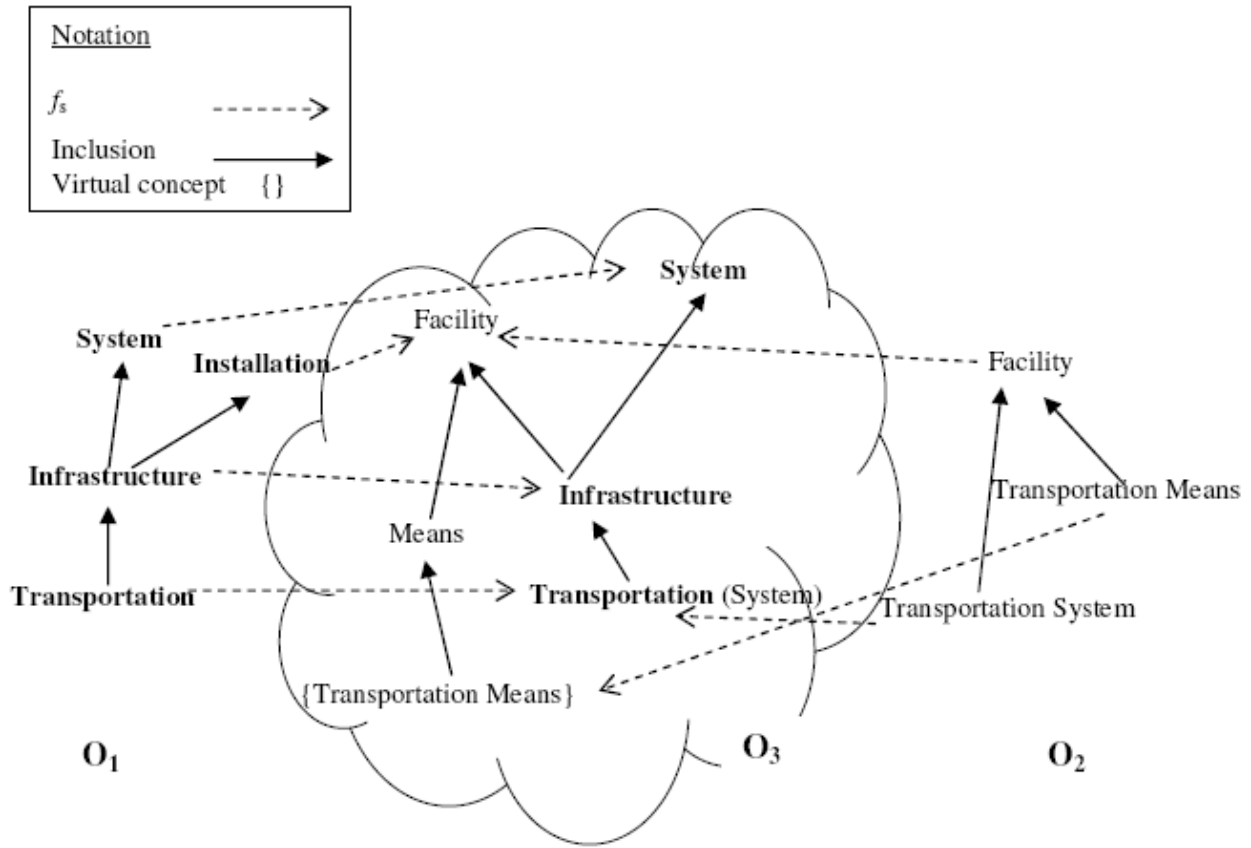


Figure 4: A cycle flow of semantic information across ontologies

In figure 4 we consider O_3 to be part of a larger intermediate ontology and so define the mapping of ontologies O_1 and O_2 by means of morphism $f_1: S_1 \rightarrow S_3$ and $f_2: S_2 \rightarrow S_3$, i.e. by means of their mapping to the intermediate ontology. The intermediate ontology here in our case is the WordNet.

2.3.1 What is ontology matching?

Ontology matching can be defined as the process of discovering similarities between two ontologies, and it can be processed by exploiting a number of different techniques. i.e. similar entities or subsumption. The result of a matching process is called mapping and can be defined as follows:

To map an ontology O_1 onto an ontology O_2 implies that for each entity (concept or relation) of O_1 a corresponding entity of O_2 having the same meaning is searched. Formally, the matching function performing this process can be defined as follows:

$$\text{match} : (O_1, O_2) \rightarrow \{(c_1, c_2), \text{with } c_1 \in O_1, c_2 \in O_2 \text{ and } \text{sim}(c_1, c_2) > t\},$$

whereas **sim** is an arbitrary similarity function and **t** a predefined similarity threshold.

2.3.2 Why is ontology matching needed/interesting?

Ontology matching is interesting because it focuses on the key type of problems that hinder the combined use of independently developed ontologies. We identify two levels at which matching may appear:

Language or meta-model level

This describe the level of the language primitives that are used to specify an ontology. Mismatches at this level are between the mechanisms to define classes and relations.

Ontology or model level

This describes the level of the actual ontology of a domain. Mismatch at this level is a difference in the way the domain is modelled.

Our own approach is on both levels. The important note is that labels in this classification hierarchy is used to define the set of documents that we can classify under the node holding the label. This is so because while semantics associated to a label are the real world semantics, the semantics of the concept of a label are in the space of the document. The relation here is that a document should be an extension of the concept of a label so that concepts of labels can now depend on the labels themselves and are independent of where in the tree they are structured.

2.3.3 What is semantic interoperability, why is it needed?

The basic challenge in building the semantic web is enabling the interoperability among different ontologies. Ontologies can interoperate only if correspondences between their elements have been identified and established. Today, if two ontologies need to interoperate, the mapping is mainly achieved by numerous human intervention. This task is tedious, error-prone, and time consuming. The solution of semantic interoperability problem could be a bottleneck in building a network of cooperating information management systems. Hence, the introduction of new methodologies that support the knowledge engineer in discovering semantic correspondences is crucial to the success of the semantic web.

2.4 Review of ontology matching techniques

2.4.1 Glue

This is a matching technique introduced by Dram *et al.*(2004) that uses ontology instances as input. It uses machine learning techniques to collect meaningful information about the content or the syntactical representation of concepts helping to classify instances. The classification of the instances is then used to calculate a joint probability distribution using an appropriate similarity function, e.g. the Jacquard similarity, the distribution is transformed into a similarity value which in turn is transformed into a similarity matrix. The matrix, domain-specific constraints and heuristic knowledge are applied to this technique so as to obtain a matching pattern. Finally, relaxation labeling which is technique of assigning labels to the node of a graph (a graph algorithm) is used to determine an alignment. This approach differs from my own because the type of input needed depend on the available trained data.

2.4.2 Coma++

COMA by Do *et al.*(2002) is a system which flexibly combines many different matchers, e.g. the comparison of names or data types or structural matcher. In Engmann and Mabmann(2007)two instance-based matchers are proposed as an extension of the existing matcher library. The new instance-based matchers can be divided into constraint based and content-based matching methods. The constraint based approach classifies instances using general or pattern constraints. Among the general constraints there are methods calculating the average length or determining the set of the used characters. Numerical constraints check whether an instance is a number and determine its type, and pattern constraints test whether all instances belonging to one concept follow a certain pattern. The content-based algorithm compares the instances pair-wise using string-similarity functions like the edit distance. This approach is similar to my own work in the case that it uses a predefine combination of words represented in textual format in WordNet collections.

2.4.3 Automatch

This is a system by lacher and Groh(2001) that also uses matching based on some classifying instances. Data sets are created containing all possible values and their probability of occurrence. The instances of an ontology are matched against the data set, and the resulting mapping provides the basis for calculating an individual match score for each attribute. These scores are summed up so as to construct a mapping by applying a “minimum cost maximum flow” graph algorithm. This technique has the similarity of graphical representation to my own work only that it uses the concept of probability distribution to compute occurances of some values where as in my case formula is assigned to this nodes during the conversion of these textual representation to XML.

2.4.4. Duma

Dumas introduced by Bilke and Nauwmann(2005) is another instance-based matching approach. This technique tends to compare set of instances so as to identify duplicates. The information that is obtain from this technique is used to find attributes that have similar pairs and concepts. If there are duplicates, this algorithm should work very well. In real-life datasets there should be duplicates, but if there are none the algorithm does not work, just as if there are no instances available at all.

2.4.5 Scalable knowledge composition

This is a technique that uses algebra for ontology composition. It operates on directed label graphs where each operator has an input graph of semi-structured data and transforms it to a graph (composable). These operations are knowledge driven using articulation rules that include:

- Logical rules (semantic implication between terms)
- Functional rules (conversion between terms across ontologies)

The technique provides intersection operations which produces articulated ontologies that contain terms that are related and their relations. The main drawback of these technique is that rules are specified manually.

2.5 Comparisons of ontology matching techniques

The machine learning techniques used by GLUE do not work very well if instances differ syntactically. Another disadvantage is that GLUE cannot determine a mapping if there is a lack of instances. The instance-based methods of COMA++ seem to be promising, but unfortunately the developers did not tell something about the use of similarity measures. But the application of

an appropriate similarity function is very important, since the comparison of the constraints for example is very challenging. A lot of questions have to be answered, for instance if similar average length influences the similarity of two attributes. One big drawback of Automatch is the huge effort needed to build the data sets and especially to calculate the probabilities.

2.6 Limitations of ontology matching techniques

Ontology matching process is still a key research issue. Considering all the above mentioned techniques, the following limitations apply to all techniques:

1. There are always maximum sets of human decisions present in the matching process: this is because labels are often written in some not well defined subset of natural language and as we all know, natural language presents lots of ambiguities and complications. For instance many ways exist for stating the same concept as in the examples trees in Figures 5 and 6 Nigerian and Nigeria, though being different words, for our purposes, they have the same classification role.

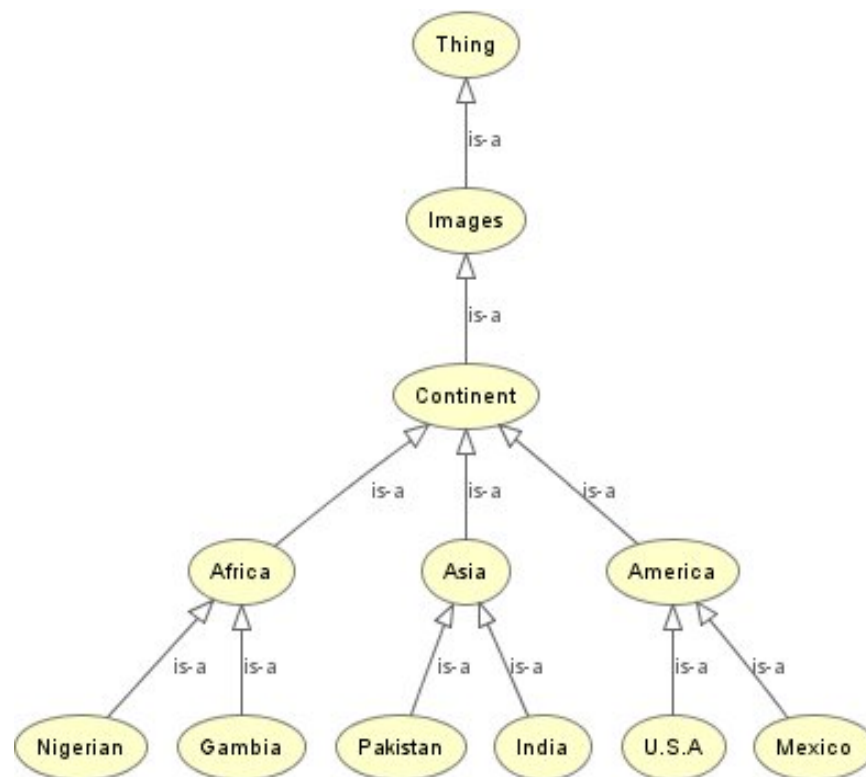


Figure 5: An image ontology domain

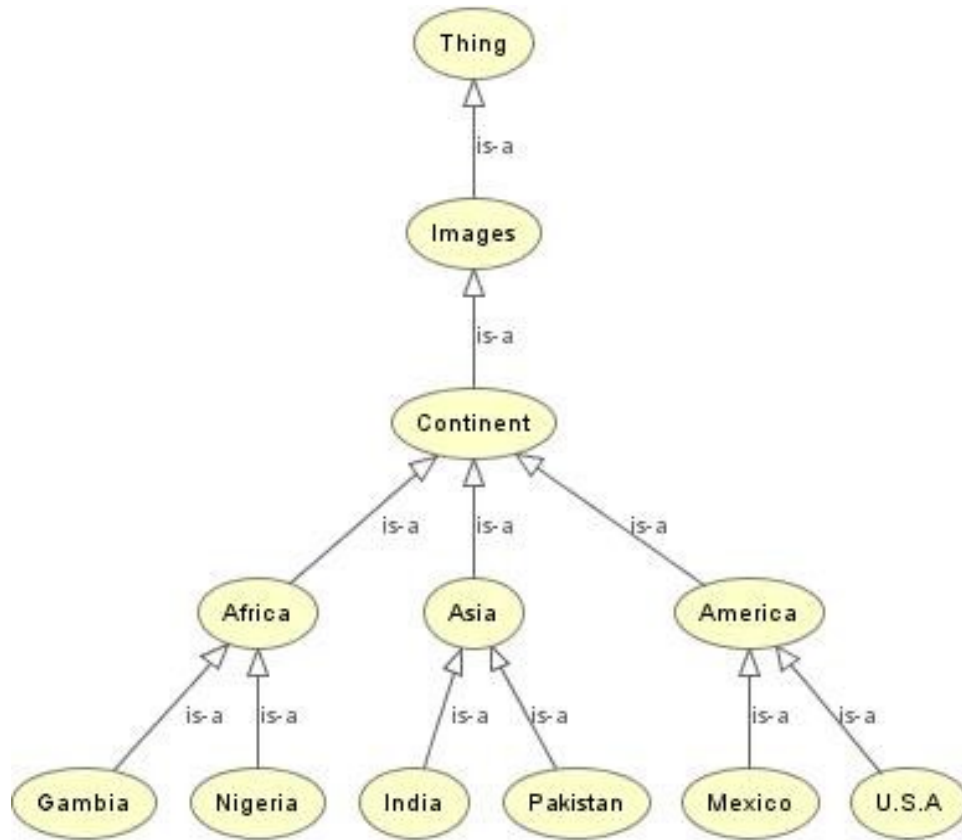


Figure 6: An image ontology domain

2. They require human involvement in the final stages of the process, for the users to verify the results and specify further mappings.

CHAPTER THREE

SYSTEM DEVELOPMENT

In this chapter, we develop a framework for reasoning about the semantic web. We start with a basic graphical approach and gradually refine it to deal with the problems of representing knowledge on the Web. In the process the following issues are discussed: Our system requirement, textual representation of graphs and patterns in graph.

3.1 System Requirement

A requirement for the semantic web is the ability to associate explicit meaning with the content of resources. We do this by embedding a logical language in the resources and providing a denotational semantics for it.

Ontology development is an engineering discipline whose development requires pre-defined stages just like software engineering. Building tools are also needed along the way. Development projects often involve solutions using numerous ontologies from external sources as well as existing and newly developed in-house ontologies. Ontologies from any source may progress through a series of versions. In the end, careful management of this collection of heterogeneous ontologies has become necessary so as to keep track of them. When starting out on an ontology project, the first and reasonable reaction is to find a suitable ontology software editor. We chose protégé software tool because we can access all of its parts through a uniform GUI (graphical user interface) whose top-level consists of overlapping tabs for compact presentation of the parts and for convenient co-editing between them. This "tabbed" top-level design permits an integration of:

1. The modeling of ontology of classes describing a particular subject.
2. The creation of a knowledge-acquisition tool for collecting knowledge.

3. The entering of specific instances of data and creation of a knowledge base.
4. The execution of applications.

First we must define our domain of discourse. The main objects of interest are internet resources and entities that are described by them. An internet resource is anything that provides information via the Internet, such as a web page, newsgroup, or e-mail message. We will use R to refer to the set of these resources. The domain of discourse, on the other hand, is the collection of things that are described or mentioned by internet resources, including potentially internet resources themselves. We will use D to refer to this set.

3.1.1 Protégé Ontology Software Tool

Ontologies may be derived from or transformed into forms such as W3C XML Schemas, database schemas, and UML to achieve integration with associated enterprise applications. Still other tools can help acquire, organize, and visualize the domain knowledge before and during the building of a formal ontology. Despite the immaturity of the field, or perhaps because of it, we were able to identify a surprising number of ontology creating tools and settle on the use protégé tool, S-Match.

In this context we use the protégé ontology tool to create our own class hierarchy for a course ontology as in the Figure 7.

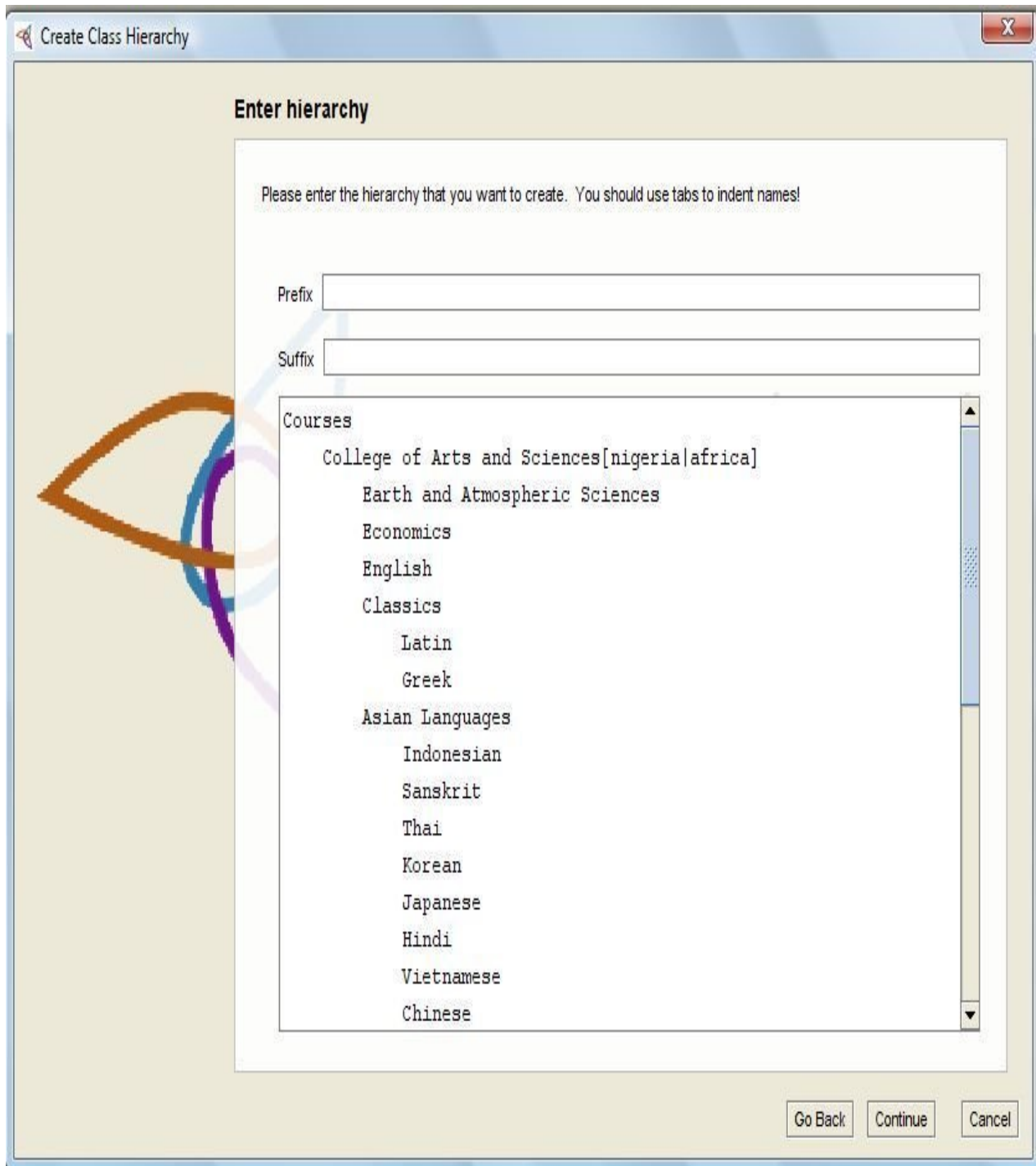


Figure 7: Class hierarchy for our course ontology using protégé tool

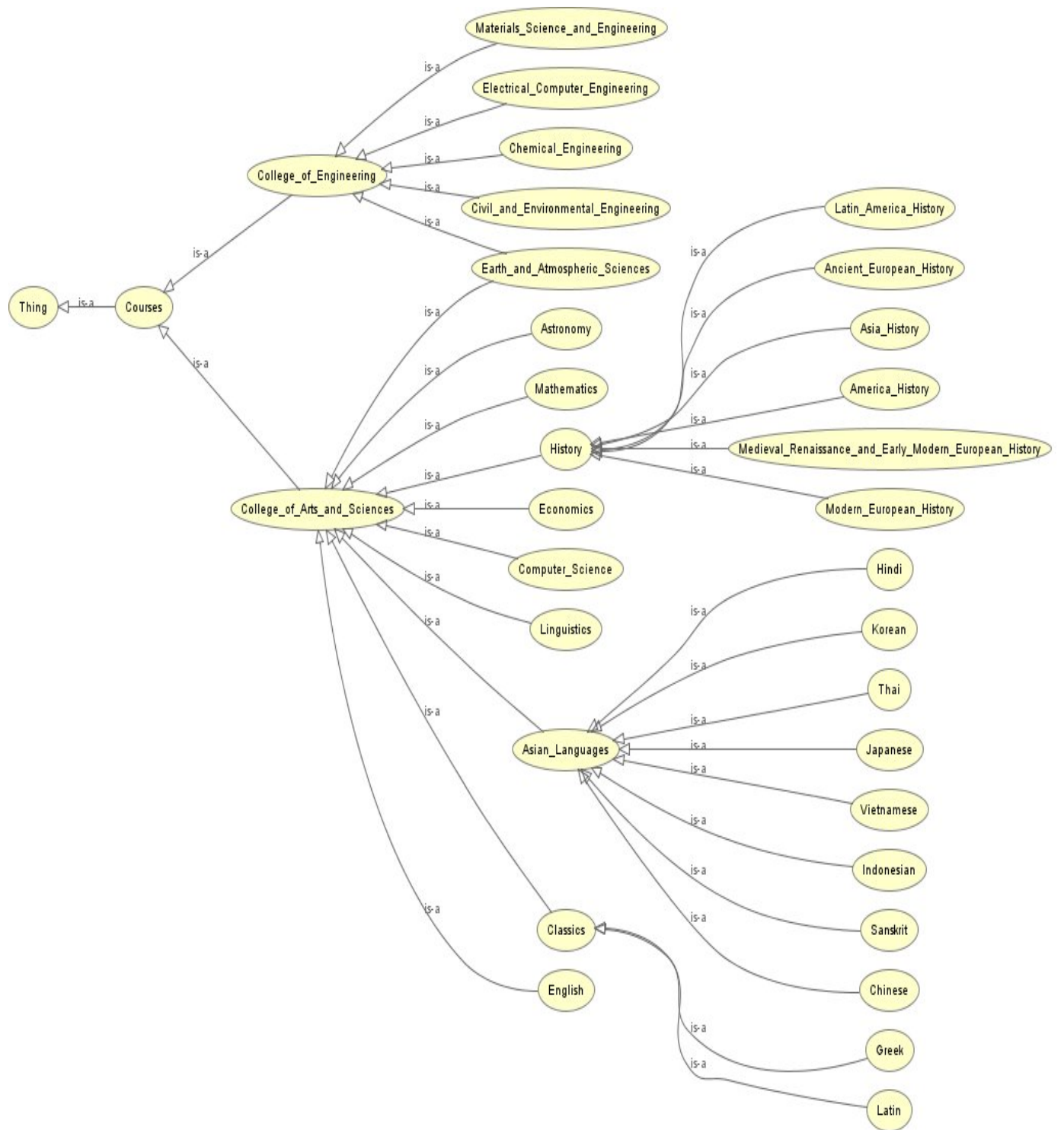


Figure 8: Inferred model for our course ontology using protégé tool

This structure in Figure 8 above will serve as the input to the S-match tool. Since graphs like this are not machine interpretable, we use our textual representation that are used for the creation of class hierarchy as inputs which is then converted to XML trees.

The following are some examples of class hierarchies that we have created with different number of nodes and labels so as to be able to test our proposed enhanced implementation.

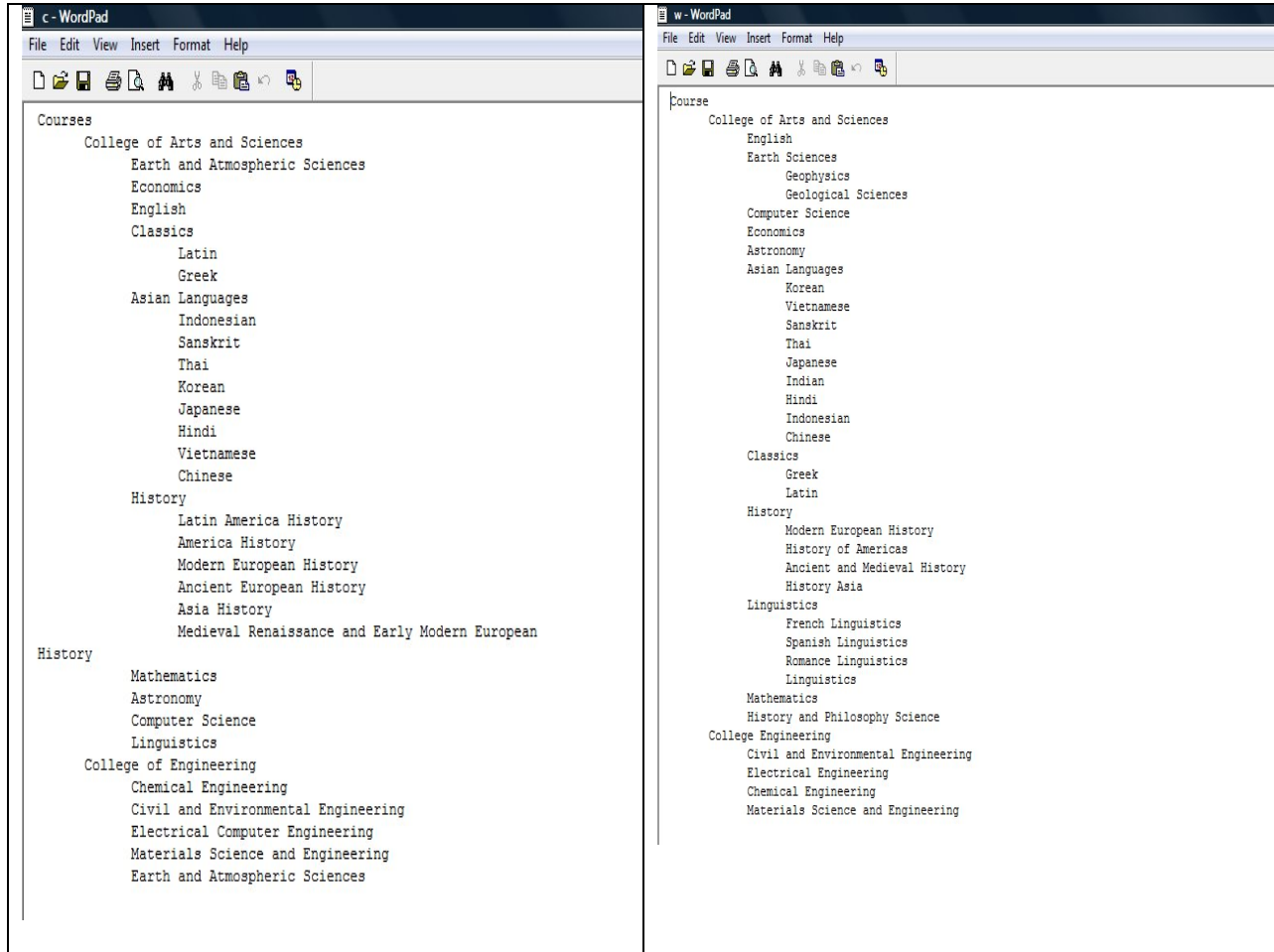


Figure 9: Class hierarchy for two different course ontologies

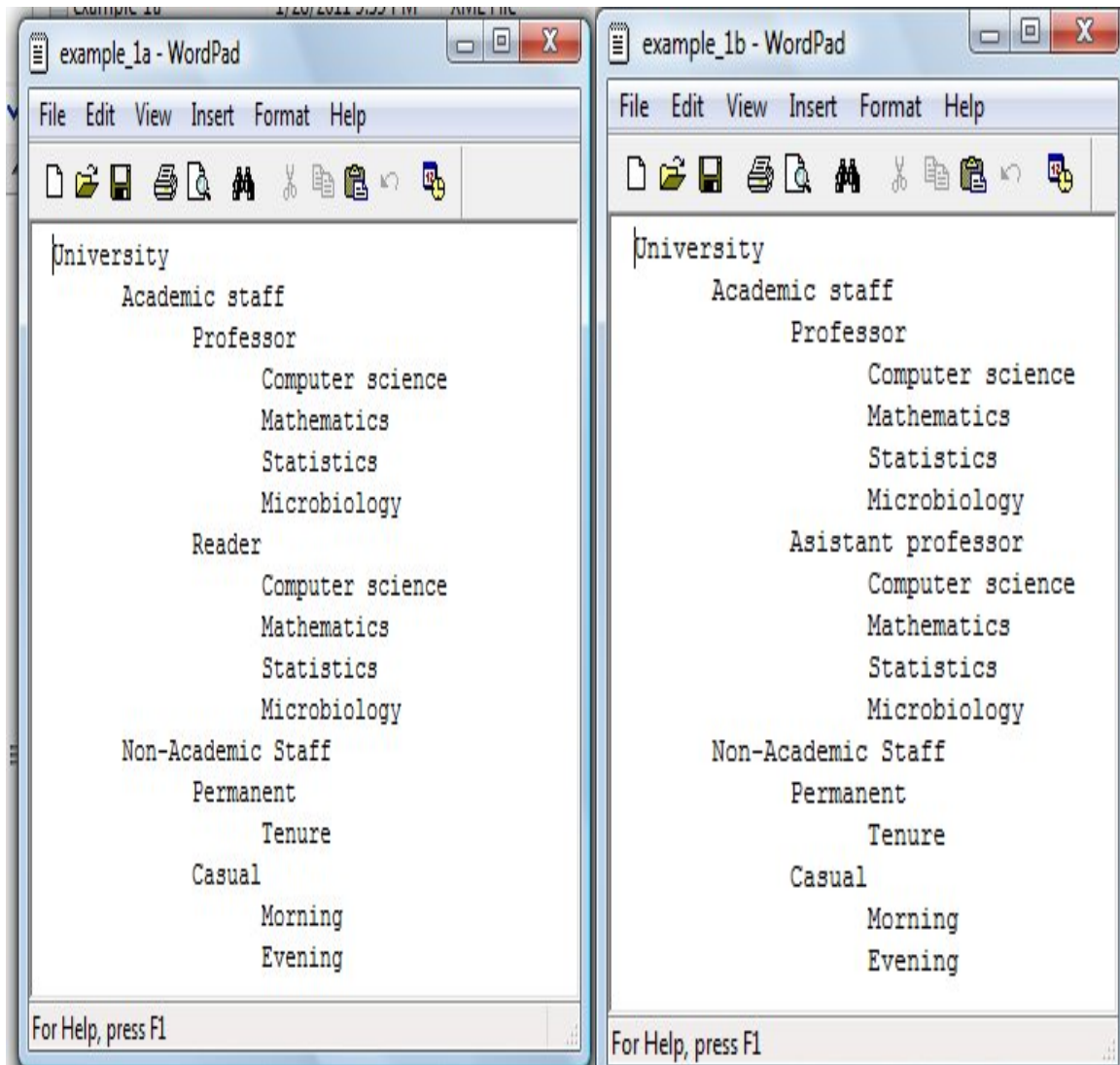


Figure 10: Class hierarchy for two different university ontologies

The above textual representation represent two different class hierarchies of university domain.

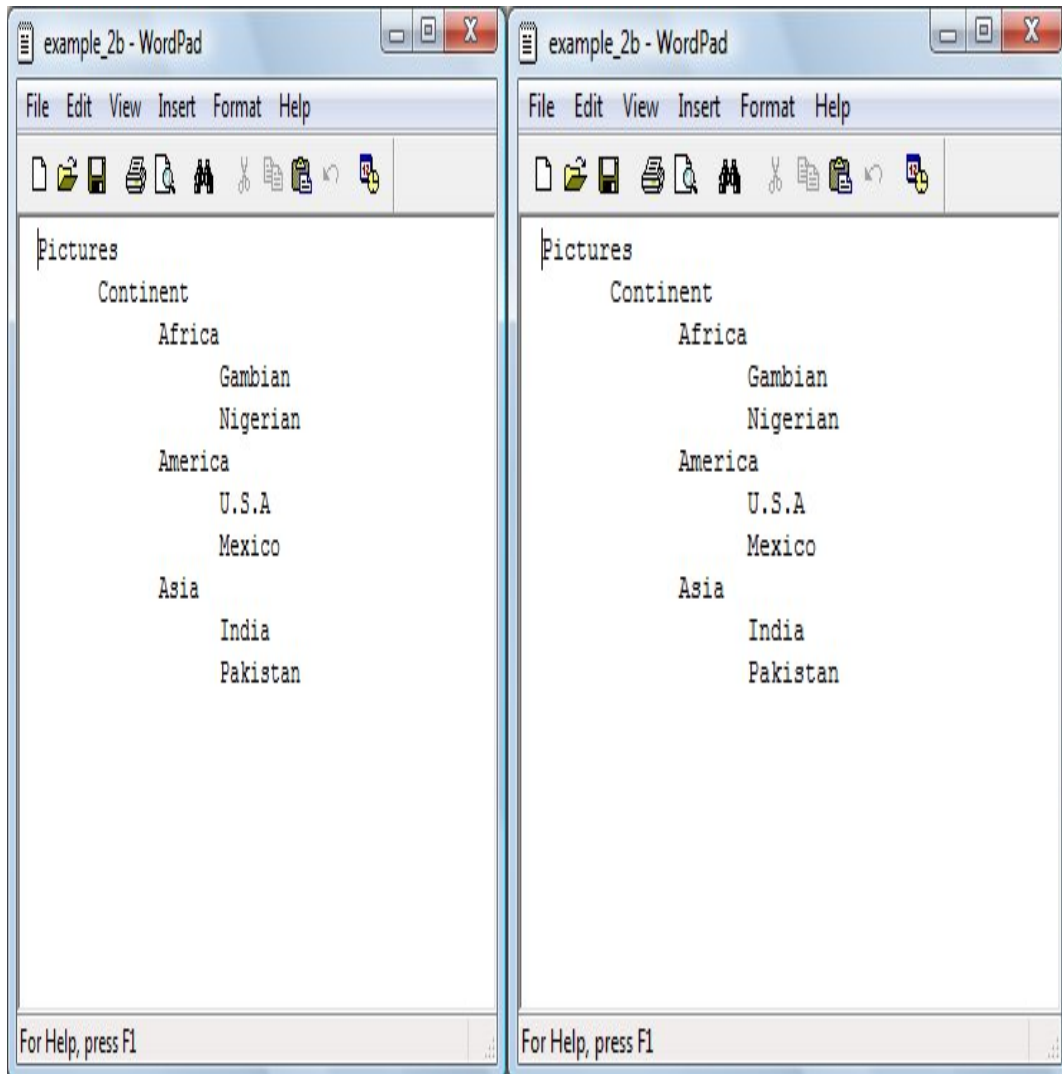


Figure 11: Class hierarchy for two different picture ontologies

The textual representation of two different class hierarchies of a picture domain ontology. As discussed earlier these are the inputs to our systems.

3.1.2 S-match

S-Match is a semantic matching algorithm developed at the University of Trento. It is a research software, intended mainly for use by researchers to conduct experiments. The release contains not very robust and not optimized code. Its main use was and is to conduct experiments. The

excellent aspect of this tool is that it is free and open source, that means we can make modification to these tool to suit our own approach.

3.2 Introduction to graphs

Graphs are used to represent data and processes in many fields of science such as biology, engineering, and sociology. To analyze graph structured data and to uncover important properties analyzing patterns in graphs (also called motifs in networks) play an important role. Analyzing graph patterns has many applications, including:

- Finding functional compounds in biological networks: Patterns such as “feed-forward loops” may play functional roles for information processing in gene regulatory networks .
- Determining the toxicology of substances: The toxicology or carcinogenicity of chemical substances is often based on specific substructures.
- Identifying substructures of circuits: Using graph patterns, sequential logic circuits can be separated into classes such that the classification is related to the circuit’s functional description.

Many inputs for matching, such as database schemas or classifications in ontologies can be converted into a tree. We call such inputs contexts. A context contains nodes, each of which has a parent node and children nodes. The result of the matching process is a mapping between the two contexts, source and target. The two key problems here are:

1. Ontologies, Web directories are ambiguously and partially defined which gives rise to the following issues:
 - i. Meaning of label is ambiguous (since labels are expressed in Natural Language)
 - ii. Labels may be complex sentences
 - iii. Meaning of a link is ambiguous

- iv. A lot of background knowledge is left implicit
-
- 2. The notion of matching is not well defined: many, somewhat similar notions and corresponding implementations can be found in the literature.

Graphical representation of data is one of the oldest knowledge representation formalism in the semantic network. In the semantic web, each concept is represented by a node in a graph as shown in Figure 8. Concepts are related semantically using connected arcs, which may or may not be labeled. In such a representation, meaning is implied by the way a concept is connected to other concepts. In our own approach we add meaning to the nodes of a graph so that better matching can be achieved. This is done by adding some textual annotations to this class hierarchy during the initial design of these ontologies. Figure 12 defines a new class hierarchy of our course ontology which contains additional information to the nodes of our inferred class model of the course ontology

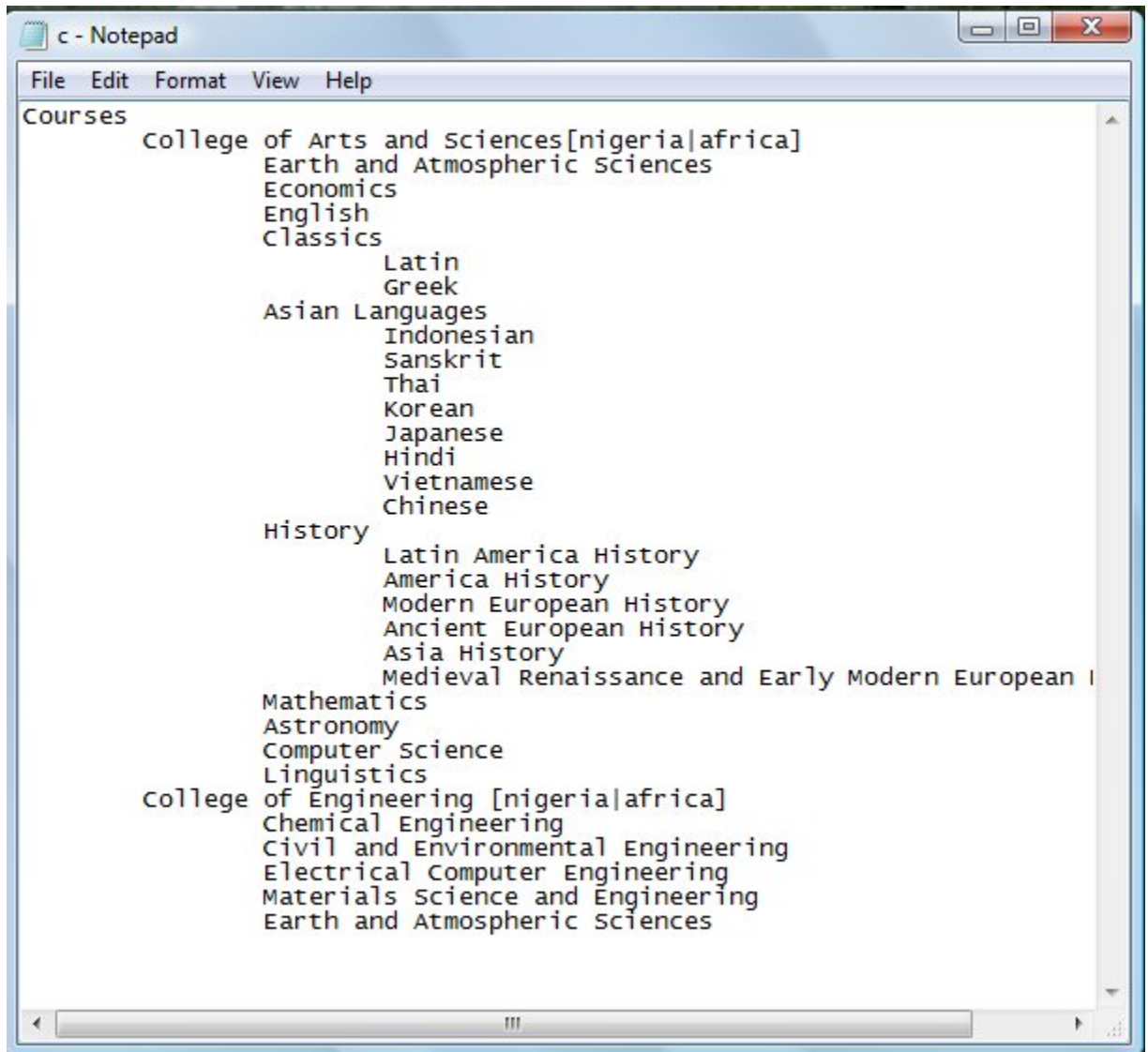


Figure 12: Class hierarchy of course ontology with textual annotations

The graphical representation of such a textual input now contain additional information as in Figure 13 below:

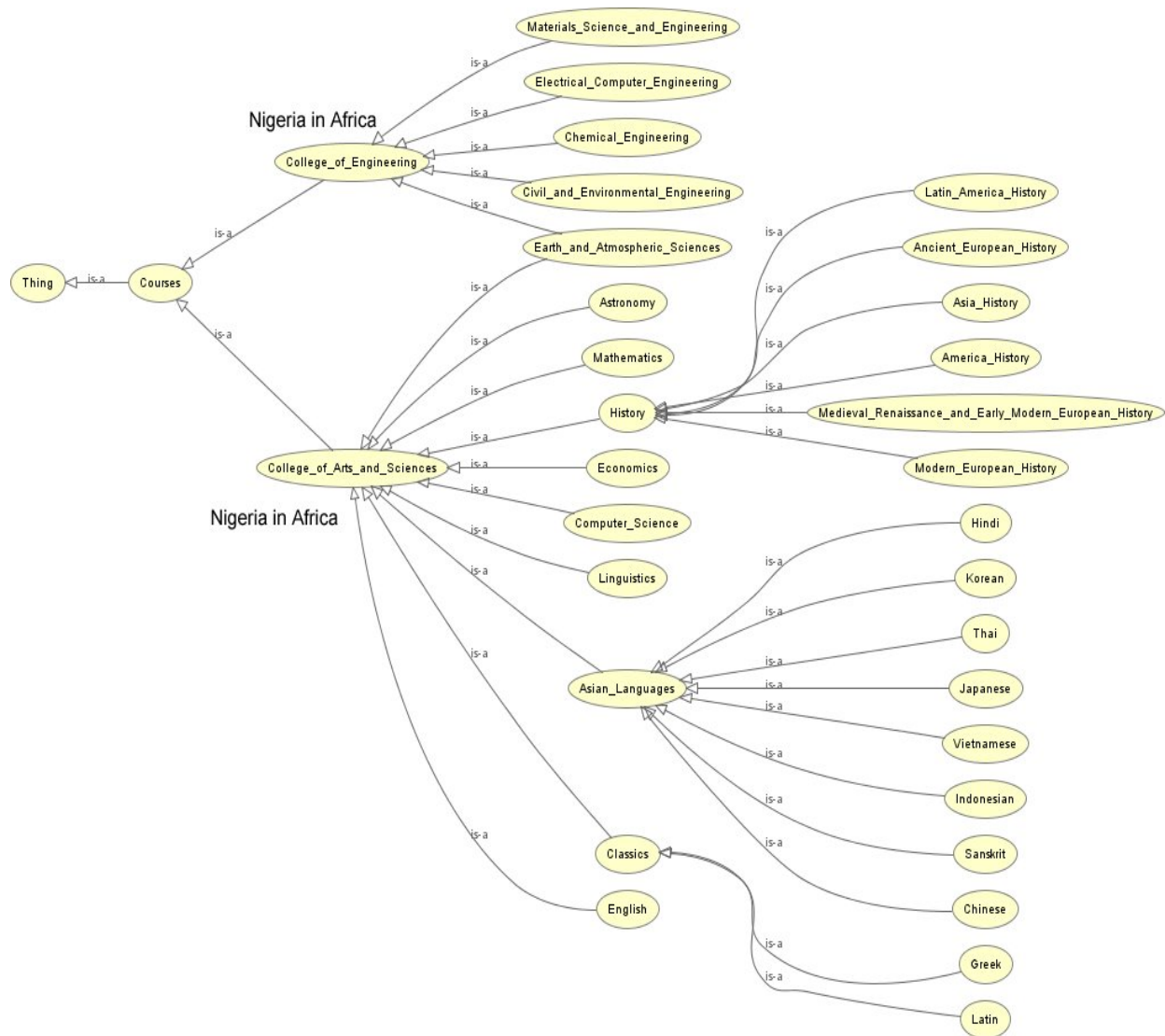


Figure 13: Tree representation of course ontology with textual annotations

3.2.1 Textual representation of graphs

In this section we introduce a method of representing nodes and edges of a graph in pure text. It is these textual representations that are our input to the matching tool. Texts are easily understood by human as we know, but to make them explicit to machines, these textual representations are now converted to graph in XML format. The algorithm works in the following pre-define stages

1. For each concept find all related attribute
2. Attach all attributes as labels
3. For all labels attached to nodes extract knowledge from WordNet.
4. The knowledge extracted is then added to the trees as annotations.

In this work we make use of S-match, which is a free and open source ontology matching tool that is built based on match tree algorithm and match tree algorithm with minimal properties.

3.3 Pattern in graphs

Patterns in our approach are labeled directed graphs. To define families of patterns we allow regular expressions as labels of vertices and edges. Every vertex or edge of a pattern may be labeled with a regular expression and therefore a single graph may specify either a single pattern, if the labels are interpreted as ordinary strings, or may specify a family of patterns.

3.3.1 Pattern specification and detection

Graph pattern matching is the problem of finding a subgraph of an input graph (the target) such that the subgraph is isomorphic to another input graph (the pattern). This type of problem is known as the subgraph isomorphism problem. In our own case we are interested in a more general graph pattern matching approach which finds multiple or all matches of a pattern in the target, this extends the scope defined by the patterns towards a more general definition of label equivalence, and restricts the matching of vertices further. Patterns are labeled directed graphs.

To define such patterns we assign additional attributes as labels of vertices and edges. Every vertex or edge of a pattern may be labeled with a textual label and therefore a single graph may specify a single pattern.

3.3.2 Pattern visualization and navigation

Research dealing with subgraph isomorphism or the detection of symmetries in graphs often considers visualization aspects. A typical visualization approach is to use force-directed layout algorithms and extend them to draw isomorphic sub graphs uniformly. However, as far as we know no work considers the problem of visualizing a graph such that sets of matches of different patterns are easily readable. In this work we present an approach that provides a layout of graphs which contains patterns where different patterns may have different layouts.

CHAPTER FOUR

SYSTEM IMPLEMENTATION

In this chapter we present an overview of WordNet application and its usage in the semantic web. We then present our algorithms for semantic matching and knowledge representation explaining how each of the algorithms works. We then discuss an implementation of our algorithm. The implementation is tested using five sample ontologies and the result is analyzed. The enhanced algorithm has the advantage of correctness and completeness because it always computes the strongest relation among concepts. Because of the more elaborate computations performed by our algorithm, however the standard s-match algorithm runs slightly faster than our enhanced s-match algorithm.

4.1 WordNet Overview

As a project, WordNet's creation started 26 years ago in 1985. WordNet is a semantic dictionary. It was designed to represent words and concepts as an interrelated system that is consistent with the way speakers organize their own mental lexicon. It is important to know that WordNet is different from an ordinary dictionary because it is neither a traditional dictionary nor a thesaurus, but it combines the features of both. WordNet contains *synsets* consisting of all the words that express a concept. Thus, expressions that are related to a given concept can be used by the user to look up others. WordNet does not go much beyond listing concepts in the form of synsets. However, the synsets are linked with each other by numerous semantic relations.

One use of WordNet is to preserve the distinction between the level of semantics and the level of concepts by enforcing different lexical relations for words and for concepts. Relations between concepts and words in WordNet are made explicit and are labeled so that users can select a specific relation to guide them from one concept to the next. Another use of WordNet is that

words express concepts in WordNet and the lexicon is constrained by the kinds of concepts that are available to human beings by virtue of perception of, and interaction with the world around us.

Now that I described the main ideas and concepts that went into the development of WordNet and some of its usage, let us take a closer look at how it is used in our structure and implementation. In our work we use WordNet to extract the semantic relations that exist between concepts by computing the semantic relation that exist between their senses.

4.2 The semantic matching algorithm

We discuss the semantic matching algorithm with the help of an example, shown in Figure 14 and Figure 15.

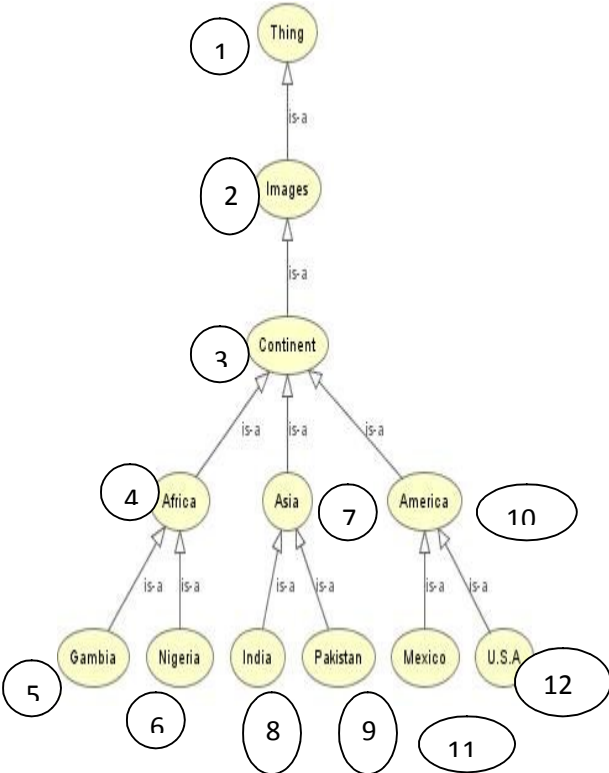


Figure 14: Tree A

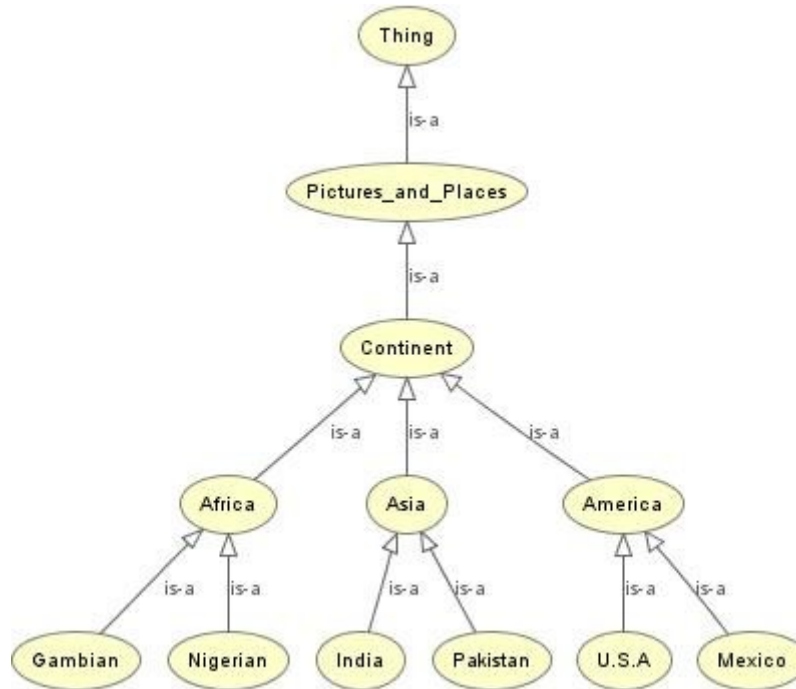


Figure 15: Tree B

Here, numbers are the unique identifiers of nodes. We use “C” for concepts of labels and concepts at nodes. For instance, in the tree A, C_{images} and C_2 are respectively, the concept of the label images and the concept at node 2. To simplify the discussion, whenever it is clear from the context we assume that the concept of a label can be represented by the label itself. In this case, for example, C_{images} becomes denoted as images. Finally, we sometimes use subscripts to distinguish between trees in which the given concept of a label occurs. For instance, Images_A , means that the concept of the label Images belongs to the tree A.

Our proposed algorithm is based on $S\text{-Match}$ which is a semantic matching algorithm. $S\text{-Match}$ takes two input files which contain tree-like structures and computes the semantic relation for each pair of nodes between the two trees using the following steps.

- Step 1 : for all labels L in two trees, compute concepts of labels, C_L .

- Step 2 : for all nodes N in two trees, compute concepts at nodes, C_N .
- Step 3 : for all pairs of nodes in two trees, compute relations among C_N 's.

Our computation is performed in five steps(Bagiwa and Junaidu, 2011):

- Step 1 : for all labels L in two trees, compute concepts of labels, C_L .
- Step 2 : for all nodes N in two trees, compute concepts at nodes, C_N .
- Step 3: Dealing with ambiguity and concept sense discrimination
- Step 4 : for all pairs of labels in two trees, compute relations among C_L 's.
- Step 5 : for all pairs of nodes in two trees, compute relations among C_N 's.

The first step is the computation of the concept of a label. The most natural choice is to take the label itself as a placeholder for its concept. Here it is important to notice that steps 1 and 2 can be done once for all, for all trees, independently of the specific matching problem and are called the off-line part. They constitute a phase of preprocessing. Step 3 remove ambiguous words and phrases. Steps 4 and 5 can only be done at run time, once the two graphs which must be matched have been chosen. Step 4 produces a matrix, called C_L matrix, of the relations holding between concepts of labels, while step 5 produces a matrix, called C_N matrix, of the strongest relations holding between concepts of nodes. These two matrices constitute the main output of our matching algorithm.

Let us consider these four steps in detail, analyzing in turn the preprocessing phase, the computation of the C_L matrix and, finally, the computation of the C_N matrix.

4.3 The tree matching algorithm: Step I Computing concepts at labels

As discussed earlier the first step is the computation of the concept of a label. The most natural choice is to take the label itself as a placeholder for its concept. After all, for instance, the label `Images` is the best string which can be used with the purpose of characterizing all documents which are about Image. We now take labels to stand for their concepts when computing the concepts of nodes. The key idea underlying semantic matching is that labels, which are written in some external language, should be translated into an internal language, the language used to express concepts. The internal language should have precisely defined syntax and semantics, thus avoiding all the problems which relate to the problem of understanding natural language. We have chosen, as internal language, a logical propositional language where atomic formulas are atomic concepts, written as single words, and complex formulas are obtained by combining atomic concepts using the connectives of set theory. These connectives are the semantic relations introduced in later sections. The semantics of this language are the obvious set-theoretic semantics. Various work on translating natural language into more or less precisely defined internal forms has been done. The core idea is to compute atomic concepts, as they are denoted by atomic labels (namely, labels of single words), as the senses provided by WordNet. In the simplest case, an atomic label generates an atomic concept. However, atomic labels with multiple senses or labels with multiple words generate complex concepts.

The following defines the steps taken to build these contexts:

1. Tokenization. Labels at nodes are parsed into tokens by a tokenizer which recognizes punctuation, cases, digits, etc. Thus, for instance, `Pictures` and `Places` becomes `<Pictures, and, Places>`, `images` becomes `<images>`.

2. Lemmatization. Tokens at labels are lemmatized, namely they are morphologically analyzed in order to find all their possible basic forms. Thus, for instance, `Images` is associated with its singular form, `Image`.
3. Building atomic concepts. WordNet is queried to extract the senses of lemmas at tokens identified during step 2. For example, the label `Images` has the only one token `Images`, and one lemma `Image`, and from WordNet we find out that `Image` has eight senses, seven as noun and one as a verb.
4. Building complex concepts. When existing, all tokens that are prepositions, punctuation marks, conjunctions (or strings with similar roles) are translated into logical connectives and are used to build complex concepts out of the atomic concepts built in step 3 above. Thus, for instance, commas and conjunctions are translated into disjunctions, prepositions like `of`, `in` are translated into conjunctions, and so on. For instance, the concept of label `Pictures and Places`, $C_{\text{Pictures and Places}}$ is computed as $C_{\text{Pictures and Places}} = \langle \text{Pictures}, \{\text{sensesWN}\#2\} \rangle \langle \text{Places}, \{\text{sensesWN}\#1\} \rangle$ where $\langle \text{Places}, \{\text{sensesWN}\#3\} \rangle$ is taken to be union of the four senses that WordNet attaches to `Pictures`, and similarly for `Places`.

After the first phase, all labels have been translated into sentences of the internal concept language. The tree has been added with more explicit knowledge.

The goal of the second step is to compute concepts at nodes. These are written in the same internal language as concepts of labels and are built suitably composing them. In particular, the key observation is that a document, to be classified in a certain node, must be in the extension of the concepts of the labels of all the nodes above the node, and also in the extension of the

concept of the label of the node itself. In other words, the concept C_n of node n , is computed as the intersection of the concepts at labels of all the nodes from the root to the node itself. Thus, for example, C_4 in Tree A (the node with label Africa) is computed by taking the intersection of the concepts of labels Africa, Continent, Images and Thing

$$C_4 = \text{Africa} \cap \text{Continent} \cap \text{Images} \cap \text{Thing}$$

The pseudo-code of a basic solution for the step above is given below:

```

1. concept: wff;
2. node: struct of {
3. nid: int;
4. lab: string;
5. clab: concept;
6. cnod: concept;
7. };
8. T: tree of(node);
9. function TreePrep(T){
10. foreach node  $\in$  T do{
11. node.clab := BuildClab(node.lab);
12. node.cnod := BuildCnod(node, T);
13. }}
14. function BuildCnod(node, T){
15. return Filter(mkwff( $\square$ , GetCnod(GetParent(node, T)),
16. node.clab));}

```

4.4 Dealing with ambiguity using concept sense discrimination algorithm

We have determined that ontology data and conceptual models can be represented as graphical trees. Step 1 and step 2 of our algorithm provide a means for adding more semantic information to these trees. However the semantic information are added in some natural language which may be ambiguous, We present our own method that can be used to remove such ambiguity using our concept sense discrimination algorithm.

Input: Ontology O , concept $\beta \in O$, radius r and the word w

Output: sense number of w

```

1: build an array  $N[\text{synset}(w)]$ ;
2: for each  $t1$  in  $\text{synset}(w)$ 
3: initialize  $N[\text{senseNumber}(t1)] = 0$ ;
4: for  $i = 1$  to  $r$ 
5: set  $M = \emptyset$ 
6: for each  $c$  in  $\text{scope}(\beta, i) - \text{scope}(\beta, i-1)$ 
7: set  $S = \emptyset$ ; //set of similarity values
8: for each  $t2$  in  $\text{synset}(\text{label}(c))$ 
9:  $\text{sim} = \text{similarity\_WP}(t1, t2)$ 
10:  $S = S \cup \{\text{sim}\}$ ;
11: end for each
12:
13: set  $\text{max} = \text{maximum in } S$ ;
14:  $\text{max} = \text{max} / |\text{scope}(\beta, i)|$ ;
15:  $M = M \cup \{\text{max}\}$ ;
16: end for each
17: set  $\text{Sum} = \text{summation of all the values in } M$ 
18:  $\text{Sum} = \text{Sum} / i$ ;
19:  $N[\text{senseNumber}(t1)] += \text{Sum}$ ;
20: end for
21: end for each
22:  $i1 = \text{sense number with the highest frequency in } N$ .
23: return  $i1$ 

```

First step is to declare an array structure whose size corresponds to the number of synsets (or senses) associated to the given word w that represents a concept or label in step 1 and step 2. The goal here is to maintain in each cell of the vector a pertinence value that represents how much the word w is semantically related to that sense (or belongs to that synset). The algorithm selects all the concepts in the scope of β (belonging to the reference ontology O) by varying the radius r (lines 4-6), in order to get different sets of terms. Then compute the similarity between two terms coming from the concept name of β and the word w (line 9). \emptyset is the similarity threshold value. Senses within a particular threshold are said to be ambiguous as such are treated independently and not eliminated whether they represent same entity of the domain.

4.5 The computation of the C_L matrix

At run time the first step is to compute the C_L matrix containing the relations existing between any two concepts of labels in the two trees. This step requires a lot of a prior knowledge that has been explicitly specified in step1 and 2. This distinguishes it in lexical and domain knowledge.

We use two sources of information:

1. We use a library of what is called “weak semantics element level matchers”. These matchers basically do string manipulation (e.g., prefix, postfix analysis, n-grams analysis, edit distance, soundex, data types, and so on) and try to guess the semantic relation implicitly encoded in similar words. Typical examples are discovering that P.O. and Police Officer are synonyms and the same for phone and telephone.
2. We use a library of what is called “strong semantics element level matchers”. These matchers extract semantic relations existing between concepts of labels using oracles which memorize the necessary lexical and domain knowledge (possible oracles are, for instance, WordNet, a domain ontology, a thesaurus).

Our implementation computes semantic relations interms of relation between WordNet senses.

We have the following cases:

Equivalence: one concept is equivalent to another if there is at least one sense of the first concept, which is a synonym of the second.

More general: one concept is more general than the other if there exists at least one sense of the first concept that has a sense of the other as a hyponym or as a meronym.

Less general: one concept is less general than the other iff there exists at least one sense of the first concept that has a sense of the other concept as a hypernym or as a holonym.

Mismatch: two concepts are mismatched if they have two senses (one from each) which are different hyponyms of the same synset or if they are antonyms.

For example according to Wordnet the concept denoted by label image has the first sense which is a synonym to the first sense of the concept denoted by label images, therefore image is equivalent to images.

Pseudo code implementing step 3

```
1. i, j: int;  
2. N1, N2: int;  
3. n1, n2: node;  
4. T1, T2: tree of (node);  
5. relation = {=,  $\sqsubseteq$ ,  $\supseteq$ ,  $\perp$ };  
6. clabreltemp, ClabMatrix(N1,N2): relation ;  
7. function mkClabMatrix(T1,T2){  
8. for (i = 0; i < N1; i++ ) do  
9. for (j = 0; j < N2; j++ ) do  
10. ClabMatrix(i,j)= GetReltion(T1(i).clab,T2(j).clab)}  
11. function GetRelation(clab1, clab2){  
12. clabreltemp:= GetRelFromWordNet(clab1, clab2);  
13. if (clabreltemp == “ ”)  
14. return GetRelFromMatcherLibrary(clab1, clab2);  
15. else  
16. return clabreltemp;}
```

Figure 14 and Figure 15 are the trees preprocessed by the pseudo code for step 1 and step 2. N1 and N2 are the nodes of Tree A and Tree B respectively, which represent the number of concepts of labels occurring in Tree A and Tree B respectively. *ClabMatrix* is the bidimensional array memorizing the C_L matrix. *GetRelation* tries WordNet first and then, in case WordNet returns no relation, it returns blank.

TreeA TreeB	Thing	Images	Continent	Africa	Gambia	Nigeria	Asia	India	Pakistan	America	Mexico	U.S.A
Thing	=											
Picture and Places		≡		≡	≡	≡	≡	≡	≡	≡	≡	≡
Continent			=									
Africa				=	≡	≡						
Nigerian			≡			≡						
Gambian			≡		=							
Pakistan									=			
India								=				
U.S.A												=
Asia												
America							=					
Mexico											=	

Table 1: Semantic Relation Between Concepts Of Tree A And Tree B

Each intersection between a row and a column contain a semantic relation. An empty square mean no relation have been found.

4.6 The computation of the C_N matrix

The second step at run time computes the relations between concepts at nodes. This problem cannot be solved simply by asking an oracle containing static knowledge. The situation is far more complex, being as follows:

1. We have a lot of background knowledge computed in the C_L matrix, codified as a set of semantic relations between concepts of labels occurring in the two graphs. This knowledge is the background theory/axioms which provide the context within which we reason.
2. Concepts of labels and concepts of nodes are codified as complex propositional formulas. In particular concepts of nodes are intersections of concepts of labels.

3. We need to find a semantic relation (e.g., equivalence, more general, mismatch) between the concepts of any two nodes in the two graphs.

The key idea behind this approach is to translate all the semantic relations into propositional connectives in the obvious way (namely: equivalence into equivalence, more general and less general into implication, mismatch into negation of the conjunction) and then to prove that the following formula is valid.

$$Context \rightarrow rel(C_i, C_j)$$

where C_i is the concept of node i in graph 1, C_j is the concept of node j in graph 2, rel is the semantic relation (suitably translated into a propositional connective) that we want to prove holding between C_i and, C_j , and $Context$ is the conjunction of all the relations (suitably translated) between concepts of labels mentioned in C_i and, C_j .

We take the relations between concepts at labels computed in step 3 as axioms and now we refer to them as $Context$ for reasoning about relations between concepts at nodes. We then construct the propositional formula

$$C1_i = C2_j \text{ is translated into } C1_i \leftrightarrow C2_j$$

$$C1_i \sqsubseteq C2_j \text{ is translated into } C1_i \rightarrow C2_j \text{ (analogously for } \supseteq)$$

$$C1_i \perp C2_j \text{ is translated into } \neg (C1_i \wedge C2_j)$$

In our own case we assume that a propositional formula is valid if and only if its negation is unsatisfiable.

Example

Suppose we want to check if $C1_{gambia} = C2_{gambian}$

$$\underbrace{(C1_{Continent} \Rightarrow C2_{Gambian}) \wedge (C1_{Gambia} \leftrightarrow C2_{Gambian})}_{Context} \rightarrow \underbrace{(C1_{Thing} \wedge C1_{images} \wedge C1_{Continent} \wedge C1_{Africa} \wedge C1_{Gambia})}_{C1_{Gambia}} \leftrightarrow \underbrace{(C2_{Thing} \wedge C2_{Pictures\ and\ Places} \wedge C2_{Continent} \wedge C2_{Africa} \wedge C2_{Gambian})}_{C2_{Gambian}}$$

The answer in this case is true since the relation evaluates to true from the SAT decider. Since the negation is unsatisfiable.

4.7 Discussion

The implementation was tested using five sample ontologies found on the internet. Four of the examples were developed by the semantic web consortium so as to test ontology matching while one example is developed by us. Sample one is the course ontology domain, sample two is the picture ontology domain, sample 3 is the pizza ontology domain, sample four is the university ontology domain and the last one is the family ontology domain developed by us. These examples were selected so as to ascertain the accuracy and running time of our own implementation and the S-Match implementation. Ontologies are converted into textual representations used as inputs to our algorithm. For example the ontology trees in figure 14 and 15 are represented into the textual forms shown in figure 16.

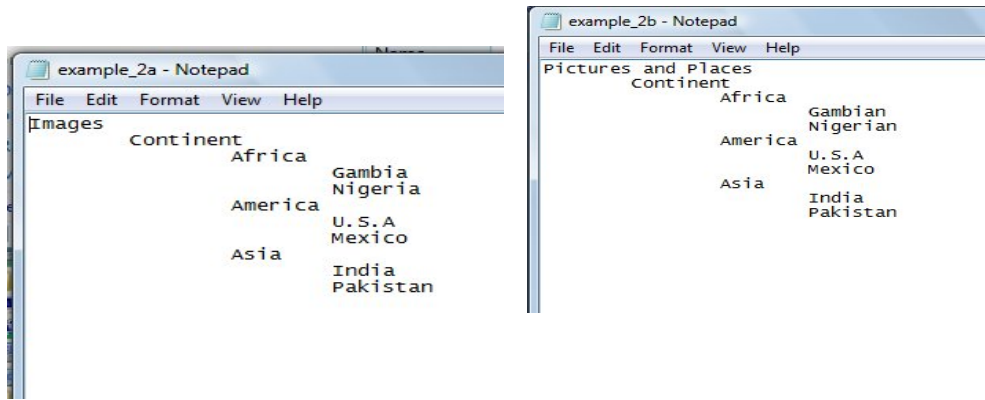


Figure 16: Sample Ontologies

These two representations are saved into two different text files input to our own algorithm. The aim here is to deduce a matching between the two ontologies not only at the lexical level both also using the knowledge residing in the two trees. Table 2 shows the outputs using S-Match and our own implementation.

Table 2: S-Match and Enhanced S-match Comparison

S/No	Inputs	Outputs										Running Time	
		S-Match					Enhanced S-Match					S-Match	Enhanced S-match
		Links	MG	LG	EQ	DJ	Links	MG	LG	EQ	DJ		
1	Sample1	135	55	65	15	0	23	7	1	15	0	8.4s	9.35s
2	Sample 2	65	26	24	8	7	16	3	1	8	4	5.98s	6.99s
3	Sample 3	223	98	93	31	1	36	4	6	25	1	10.38s	10.98
4	Sample 4	88	30	12	31	15	46	1	0	30	15	5.34s	7.38s
5	Sample 5	245	69	98	53	25	99	29	13	53	4	15.36s	19.46s

Analyzing the outputs from the two systems we can see that the number of rendered links is higher in S-match for all the example ontologies. This is because the trees in the input using S-match have no prior knowledge that describe the domains of their concepts attached as labels which make them distinct. As such, for each occurrence of a concept the s-match algorithm does not only look at the domains of that attribute but also other domains that defines similar attributes. In our enhanced s-match algorithm, the textual annotations that are added to the trees during the first and second step of our algorithm try to eliminate all dissimilar concepts. As such, the enhanced s-match algorithm only consider distinct attributes for the matching process. The concept sense discrimination algorithms try to remove ambiguous words during the matching process. The number of equivalent concepts found by both the standard s-match algorithm and our enhanced s-match algorithm is the same for all the sample ontologies. The less general and more general cases found in the standard s-match implementation is higher. However the enhanced s-match algorithm takes longer time to run because of the more elaborate computation it performs.

4.8 An Architecture of our enhanced S-match implementation

The logical architecture of the system we have developed called enhanced S-match is depicted in figure 17. Let us now discuss it from a data flow perspective.

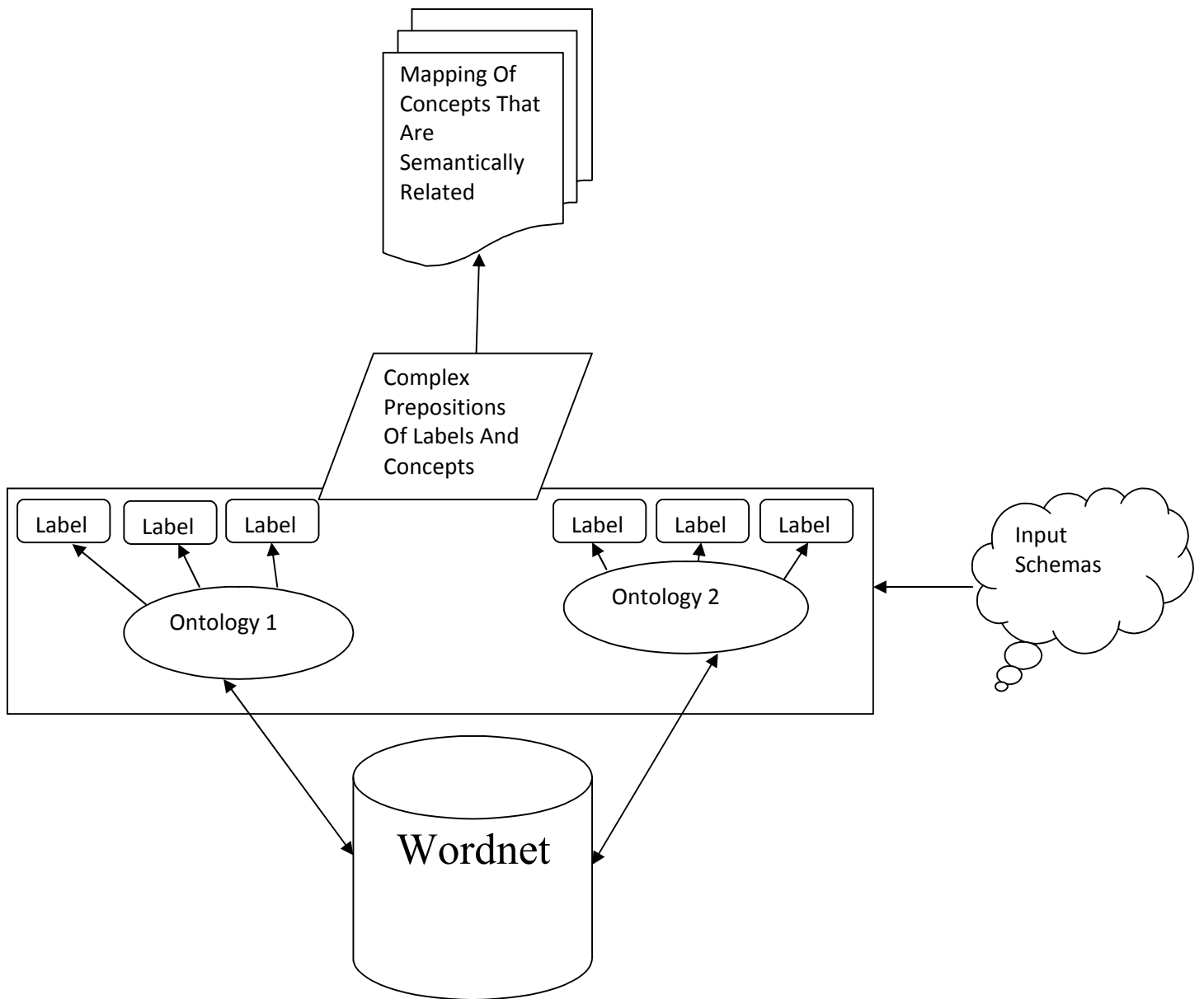


Figure 14: Architecture of the enhanced S-Match Platform

The module taking input schemas does the preprocessing. it takes in input trees codified into standard XML format. This internal format can be loaded from a file manually edited or can be produced from input dependant translator. The next module implements the preprocessing phase, enriched trees which contain concepts of labels and concepts of nodes and the produces an output. The preprocessing phase has access to the WordNet which provides the necessary lexical

and domain knowledge. Complex formulae are built from this knowledge so as to form contexts and relations between concepts which can then be solved using SAT deciders to produce a semantic mapping.

4.9 Conclusion

We have presented an enhancement of S-Match which produces a better matching accuracy, eliminates non matching concepts at an early stage and remove ambiguities that may come from natural language. There are challenges that must be addressed in order to be able to build the ultimate good-for-all semantic matcher. For example we have not eliminated the problem of incorrectness and incompleteness. We have only limited it to the translation of labels into concepts of labels (step 1) and to the computation of the semantic relations existing between pairs of concepts of labels (step 4). Steps 2 and 5, instead guarantee correctness and completeness. However, while, if step 2 is uncontroversial, step 5 still presents problems. These problems are due to the fact that, we have codified the semantic matching problem into a CO-NP problem (as it is the validity problem for the propositional calculus). Solving this class of problems requires, in the worst case, exponential time.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

This chapter presents our conclusion, directions for future work and recommendation.

5.1 Conclusion and future work

With the proliferation of data sharing applications that involve multiple ontologies, the development of automated techniques for ontology matching will be crucial to their success. This thesis presented a semantic matching technique that explores the mapping between the meanings of concept specifications by exploiting domain knowledge in two or more given ontologies. It is aimed at specifying a similarity function in the form of a semantic relation (hyperonym, hyponym, meronym, part-of, etc) between the intension (necessary and/or sufficient conditions) of concepts. Aside from striving to improve the accuracy of our methods, our main line of future research involves extending our techniques to handle more sophisticated mappings between ontologies (i.e., non 1-1 mappings), and exploiting more of the constraints that are expressed in the ontologies (via attributes and relationships, and constraints expressed on them).

5.2 Recommendation

The major recommendation in this thesis work is to take into account of a minimal number of human intervention in combining these ontologies. This can be achieved by developing a clear and concise knowledge representation domain that captures trained learners of various ontology structure specific for each application domain.

REFERENCE

1. Bagiwa A.M and Junaidu S.B. (2011). A conceptual framework for adding textual annotations to hierarchical trees representing ontologies as a means of achieving semantic interoperability between different ontologies of the same domain. *International journal of electrical, electronics and computer systems (IJEECS)*, pp. 1-6.
2. Bilke, A. and Naumann, F. (2005). Schema Matching Using Duplicates. *Proceedings of the 21st International Conference on Data Engineering, ICDE*. Tokyo, Japan, pp. 69–80.
3. Do, H. H. Melnik, S. and Rahm, E. (2002). Comparison of schema matching evaluations. *Proceedings of the workshop on Web and Databases*, pp 44-52.
4. Do, H.H. and Rahm, E. (2002). COMA - A System for Flexible Combination of Schema Matching Approaches. *Proceedings of 28th International Conference on Very Large Data Bases*, August 20-23, 2002, Hong Kong, China. Morgan Kaufmann, 2002, pp. 610–621.
5. Doan, A. Madhavan, J. Domingos, P. and Halevy, A.Y. (2004) “Ontology Matching: A Machine Learning Approach. *Handbook on Ontologies*. Springer, 2004, pp. 385–404.
6. Engmann, D. and Mabmann, S.(2007). Instance Matching with COMA++ in Database systems in Business, Technology and Web (BTW 2007), *Workshop Proceedings*, 5.-6. March 2007, Aachen, Germany pp 251-282.
7. Giunchiglia, F. and Shvaiko, P. (2003). Semantic matching. *The Knowledge Engineering Review Journal*, 18(3):265–280.
8. Gruber , T. (2009). *Ontology*, *Encyclopedia of Database Systems*. Springer-Verlag, in press.
9. Gruber, T. (1993). A translation of approach to ontology specification in knowledge acquisition 5:199-199.
10. Guarino, N and Giaretta, P. (1995). *Ontologies and knowledge bases: towards a terminological clarification towards very large knowledge bases*, Amsterdam, IOS Press, pp 25-32.

11. Lacher, M.S. and Groh, G. (2001). Facilitating the exchange of explicit knowledge through ontology mappings. Proceedings of the fourteenth international florida artificial intelligence research society conference. pp 85-92.
12. Melnik, S. Garcia-Molina, H. and Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm. Proceedings of the International Conference on Data Engineering (ICDE), pp 117–128.
13. Meta Data Coalition. Open information model, version 1.0.
<http://mdcinfo/oim/oim10.html>, August 1999.
14. Noy, N.F. and Guinness M.C. D.L. Ontology Development: A Guide to Creating Your First Ontology, Stanford Knowledge Systems Laboratory Technical Report, http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html, accessed 2nd June, 2010.
15. Wang, S. (2008). Ontology of learning objects repository for pedagogical knowledge sharing, pp. 39-59.

APPENDIX ONE

```
import it.unitn.disi.smatch.components.ConfigurableException;
import it.unitn.disi.smatch.data.mappings.IMappingElement;
import
it.unitn.disi.smatch.matchers.element.ISenseGlossBasedElementLevelSemanticMat
cher;
import it.unitn.disi.smatch.oracles.ISynset;

import java.util.Properties;
import java.util.StringTokenizer;

public class WNGlossComparison extends Configurable implements
ISenseGlossBasedElementLevelSemanticMatcher {

    private static final String THRESHOLD_KEY = "threshold";
    private int threshold = 2;

    // the words which are cut off from the area of discourse
    private static final String MEANINGLESS_WORDS_KEY =
"meaninglessWords";
    private String meaninglessWords = "of on to their than from for by in at is are
have has the a as with your etc our into its his her which him among those against
";

    @Override
    public boolean setProperties(Properties newProperties) throws
ConfigurableException {
        boolean result = super.setProperties(newProperties);
        if (result) {
            if (newProperties.containsKey(THRESHOLD_KEY)) {
                threshold =
Integer.parseInt(newProperties.getProperty(THRESHOLD_KEY));
            }

            if (newProperties.containsKey(MEANINGLESS_WORDS_KEY)) {
                meaninglessWords =
newProperties.getProperty(MEANINGLESS_WORDS_KEY) + " ";
            }
        }
    }
}
```

```

    }
    return result;
}

/**
 * Computes the relations with WordNet gloss comparison matcher.
 *
 * @param source gloss of source
 * @param target gloss of target
 * @return synonym or IDK relation
 */
public char match(ISynset source, ISynset target) {
    String sSynset = source.getGloss();
    String tSynset = target.getGloss();
    StringTokenizer stSource = new StringTokenizer(sSynset, ".,\\\"()");
    String lemmaS, lemmaT;
    int counter = 0;
    while (stSource.hasMoreTokens()) {
        StringTokenizer stTarget = new StringTokenizer(tSynset, ".,\\\"()");
        lemmaS = stSource.nextToken();
        if (meaninglessWords.indexOf(lemmaS) == -1)
            while (stTarget.hasMoreTokens()) {
                lemmaT = stTarget.nextToken();
                if (meaninglessWords.indexOf(lemmaT) == -1)
                    if (lemmaS.equals(lemmaT))
                        counter++;
            }
    }
    if (counter >= threshold)
        return IMappingElement.EQUIVALENCE;
    else
        return IMappingElement.IDK;
}
}

```