Extending an Ontology Alignment System with a Lexical Database

BY

BELLO, BATHTHAMA ALHASSAN

DEPARTMENT OF MATHEMATICS
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA.

October, 2014

Extending an Ontology Alignment System with a Lexical Database

BY

Bello, Baththama **ALHASSAN,** Bsc (UDUS)

Msc/SCIEN/05693/2009-2010

A PROJECT THESIS SUBMITED TO THE SCHOOL OF POSTGRADUATE STUDIES,
AHMADU BELLO UNIVERSITY, ZARIA

IN PATIAL FULFILMENT FOR THE AWARD OF
MASTER OF SCIENCE IN COMPUTER SCIENCE

DEPARTMENT OF MATHEMATICS,
FACULTY OF SCIENCE
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA

October, 2014

DECLARATION

I declare that the work in this project thesis entitled 'Extending an Ontology Alignment System with a Lexical Database' has been carried out by me in the Department of Mathematics. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this project thesis was previously presented for another degree or diploma at this or any other institution.

Bello Baththama Alhassan	_____	_____
Name of student	Signature	Date

CERTIFICATION

This thesis entitled "EXTENDING AN ONTOLOGY ALIGNMENT SYSTEM WITH A LEXICAL DATABASE" by Bello Baththama Alhassan meets the regulations governing the award of the degree of Master of Computer Science of Ahmadu Bello University, Zaria and is approved for its contribution to knowledge and literary presentation.

Signature ----------

---------- Date ----------

Chairman, Supervisory Committee

Signature ----------

---------- Date ----------

Member, Supervisory Committee

Signature ----------

---------- Date ----------

Head of Department

Signature ----------

---------- Date ----------

Dean, Postgraduate School

DEDICATION

This thesis is dedicated to my Parents Dr. Bello S.Y Alhassan and Dr. Ramlatu Jibir Daura who made me understand that even the largest task can be accomplished if it is done one step at a time and made me believe that I had so much strength and courage to persevere even when I felt lost. Thank you for your unconditional support in my life. My sisters and brothers, who have never left my side and are very special, my dear husband and friend for his patience at all times.

ACKNOWLEDGEMENT

In the name of Allah Most Beneficent, Most Merciful. All thanks are to him for giving me this wonderful Chance to live up to this time and enable me carry out this work.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. S.B Junaidu, for his excellent guidance, patience, and providing me with an excellent atmosphere for doing my research. You are such a wonderful motivator, even when coping seemed tough for us.

I will also like to extend my heartfelt gratitude to my supervisor Dr. A.A Obiniyi. You were very tolerant and determined to see us through.

I would also like to thank all my Lecturers, who throughout my educational career have supported and encouraged me to believe in my abilities. They have directed me through various situations, allowing me to reach this accomplishment.

I would also like to thank Mal. Abdullahi Muhammad, Mal. Aliyu, Fatsuma Jauro and Oyelade for guiding and helping me in my research.

A special thanks goes to my Head of Unit Mallam M.B Sani for his kindness and support, Aisha Ibrahim my friend and my other office colleagues Mal Hassan, Mal Suleiman, Mal Awwal and Mal Kamal for their patience and understanding.

To my friends and family Rabi'a Ahmed, Abdulghafar, Abdulghafur , Zubaidah , Abdus-shakur, Abdussamad, Fatima, Lubabatu, Husna, Abubakar, Jibril, Awwal, Zainab, Nafisa, Mustapha, Juwerriya, Abdullahi thank you and may Allah bless you all Amin.

ABSTRACT

Ontology alignment is a challenging problem in many applications and is a major concern for interoperability in information systems. It aims to find semantic correspondences between a pair of input Ontologies, which is still an active area of a research. One product of Ontology alignment research is Falcon aligning ontologies (Falcon-AO), a system that attempts to align a pair of input ontologies automatically. This thesis investigated the theoretical and practical aspects as well as the limitations of Falcon-AO in both aspects. An enhanced architecture of Falcon-AO was also developed and implemented. The aim was to enhance the precision of Falcon-AO's alignment through the use of a large lexical database "WordNet" in the alignment process. The extended Falcon-AO was built and validated using standard benchmark Ontologies. A comparative performance analysis of the original and extended Falcon-AO systems, based on alignment precision and execution time, shows the extended system to be more precise, although at a small execution time penalty.

TABLE OF CONTENTS

DECLARATION iii
ACKNOWLEDGEMENT vi
ABSTRACT vii
CHAPTER ONE 1
GENERAL INTRODUCTION 1
1.1 Background of the Study 1
1.2 Motivation and Problem Statement 4
1.3 Research Questions 8
1.4 Aims and Objectives 8
1.5 Goal 8
1.6 Methodology 8
1.7 Contribution to Knowledge 9
1.8 Thesis Organization 9
CHAPTER TWO 10
LITERATURE REVIEW 10
2.1 ONTOLOGY 10
2.1.1 Ontology components 11
2.1.2 Ontology languages 12
2.1.3 Ontology applications 15
2.1.4 Ontology problems 16
2.1.5 Definition 17
2.2 SEMANTIC WEB 18
2.3 ONTOLOGY MATCHING TECHNIQUES 19
2.3.1Element-level techniques 19
2.3.2 Structure-level techniques 20
2.3.3 Use of External Resources 21
2.4 REVIEW OF ONTOLOGY MATCHING SYSTEMS 22
2.4.1 Anchor-Flood 22
2.4.2 AROMA (Association Rule Ontology Matching Approach) 24
2.4.3 QOM (Quick Ontology mapping) 24
2.4.4 TaxoMap Matching Technique 26

2.4.5 ASMOV (Automated Semantic Matching of Ontologies with Verification).....27
2.4.6 FALCON-AO.....28
CHAPTER THREE.....31
DESIGN OF EXTENDED FALCON-AO.....31
3.1 System Requirement.....31
3.2 Falcon-AO.....33
3.3 Extended Falcon-AO.....33
3.3.1 Synonym Search Component.....34
3.3.2 Extended Falcon-AO System Flow.....35
3.4 OTHER COMPONENTS OF THE EXTENDED FALCON-AO.....38
3.4.1 Model pool.....38
3.4.2 Matcher library.....38
3.4.3 Alignment set.....47
3.4.4 Central controller.....47
3.4.5 Repository.....47
3.4.6 Similarity combination strategy.....48
CHAPTER FOUR.....50
IMPLEMENTATION OF THE EXTENTEDED FALCON-AO.....50
4.1 Differences of Falcon-AO and Extended Falcon-AO.....50
4.2 Processes of the Extended System.....51
4.2.1 Timer.....51
4.2.2 WordNet Processing.....51
4.2.3 Getting Synonyms from WordNet.....52
4.2.4 How Synonyms Are Retrieved From WordNet.....53
4.2.5 Implementing Same AS rule.....54
4.2.6 Getting Equivalent Entities.....54
4.2.7 How Matched Entities Are Mapped.....55
4.2.8 Merging WordNet and Falcon-AO Matched Entities.....56
4.3RESULTS AND DISCUSSION.....56
4.3.1OAEI (Ontology Alignment Evaluation Initiatives).....56
4.4.2 Experimental setup.....56
4.4 EVALUATION CRITERIA.....57
4.4.1 Precision.....57
4.4.2Time.....57

4.5 Discussion of Results 59
4.6 Summary 59
CHAPTER 5.....60
SUMMARY, CONCLUSION AND RECOMMENDATION.....60
5.1 Summary and Conclusion.....60
5.2 Limitations and Future work.....61
APPENDIX A 65

LIST OF TABLES

Table 1.1 Semantic Relations in WordNet 7

Table 2.1 Ontology Matching Techniques Classification 22

Table 2.2 Comparison of some Matching Systems.....30

Table 3.1 Some Classes and Interfaces of Jena with their descriptions 32

Table 4.0 Results of the Experiment. 58

LIST OF FIGURES

Figure 1.1 Matching Process 3

Figure 1.2 Two different Ontologies describing an article 7

Figure 3.1 Architecture of Falcon-AO 33

Figure 3.2 Architecture of Extended Falcon-AO 34

Figure 3.3 System Flow of Extended Falcon-AO 37

Figure 3.4 Ontologies of the same Domain with Different Structures 39

Figure 3.5 Overview of PBM Approach 40

Figure 3.6 Sample Ontology 1 with values of structural proximity 42

Figure 3.7 Sample of cluster from Ontology 1 42

Figure 3.8 Sample of a block from Ontology 1 42

Figure 3.9 Sample of a block from Ontology 2 42

Figure 3.10 RDF bi-partite graph 46

Figure 3.11 Directed bi-partite graph 47

Figure 4.1 Extended Falcon-AO Interface 50

Figure 4.2 Falcon-AO Interface 50

Figure 4.3 Timer Code 51

Figure 4.4 WordNet Processing 51

Figure 4.5 Getting Synonyms from WordNet 52

Figure 4.6 How Synonyms are Retrieved from WordNet 53

Figure 4.7 Implementing SameAs Rule 54

Figure 4.8 Getting Equivalent Entities 54

Figure 4.9 How Matched Entities are Mapped 55

Figure 4.10 Merging WordNet and Falcon-AO matched entities 55

ABBREVIATIONS AND DEFINITION

RDF: Resource Description Framework

RDFS: Resource Description Framework Schema

OWL: Web Ontology Language

SYNSET: Synonym Set

PBM: Partition-Based Block Matching

GMO: Graph Matcher for Ontologies

V-Doc: Virtual Documents

XML: eXtensible Markup Language

Semantics: is an individual's interpretation of data according to his/her understanding of the world.

Interoperability: is the ability of two or more systems to exchange information and to use the information that has been exchanged.

Semantic Interoperability: is the capability of different information systems to communicate information consistent with the intended meaning of the encoded information.

Information Heterogeneity: refers to the difference between information. Information heterogeneity can happen at three levels, i.e., syntax, structure and semantics levels.

Ontology Alignment: is the process of bringing two or more Ontologies into mutual agreement, making them consistent and coherent.

Similarity: is a confidence measure between two elements in different Ontologies. The similarity is expressed in a mathematical number that typically ranges in [0..1].

CHAPTER ONE

GENERAL INTRODUCTION

This Chapter gives the introductory part of the thesis which includes the background of the study, research motivations and goals, research questions for which the thesis should provide answers to and finally the summary of the thesis contribution to knowledge.

1.1 Background of the Study

The World Wide Web (WWW) is widely used as a universal medium for information exchange, because it holds vast amounts of information. Unfortunately, it is now more syntactic than semantic i.e. context and relevance. The goal is to make the content of the World Wide Web understandable and accessible to machines. Semantic interoperability among different information systems in the WWW is limited, due to information heterogeneity, and the non semantic nature of format in which the Web is defined.

The Semantic Web extends the current Web by giving it meaning, classifying and organizing information so that it is not only interpretable by humans, but also by machines. It provides methods and technologies to allow machines to understand the meaning or "semantics" of information on the WWW. To accomplish this, the provided information should be structured, accompanied by sets of inference rules that can be used by computers to conduct automated reasoning (Kaushal, 2011). These annotations could provide completely new possibilities: intelligent search instead of keyword matching, query answering instead of information retrieval, document exchange between Organizations through Ontology mappings, and definitions of views on documents.

Ontologies have been suggested as a way to solve the problem of information heterogeneity by providing formal, explicit definitions of data and reasoning ability over related concepts.

Ontologies capture the structure of the domain, as the means for conceptualizing and structuring domain knowledge, it has become the backbone to enable the fulfillment of the Semantic Web vision (Tang and Liang, 2005), this is because they are used to assist in communication between humans and computer, to achieve interoperability and communication among software agents and to improve the design quality and quality of software systems. Unfortunately, these Ontologies have their own problems of proliferation and heterogeneity, and there is need to do away with these problems by aligning the Ontologies. Multiple Ontologies often exist for overlapping domains. Ontology developers often need to integrate several Ontologies. This might be the case when they need to exchange or migrate data between different Ontology based applications, or as the first stage in the development of a new Ontology, in which relevant parts of existing Ontologies are reused.

However, since different Ontologies are typically developed by different groups of experts, even if they describe related domains, it is reasonable to expect that their vocabularies will be quite different. These need to be aligned in order to promote interoperability on the Semantic Web.

Ontology alignment is the task of identifying similar entities in different Ontologies. It can also be defined in this way according to (Hassan, *et al*, 2012).

Let O, O' be two Ontologies. Matching O with O' finds a set of alignments A= $\{a_1, a_2, \ldots a_n\}$. Each a_i (i=1, 2,..., n) is a 5-tuple: <id,e_1,e_2,u,v> where id is an unique identifier, e_1 is an entity in O, e_2 is an entity in O', u is the relationship holding between e_1 and e_2 and v is a similarity between e_1 and e_2 in the [0,1] range.

Ontology alignment is necessary to avoid overlap of information when more than one Ontologies are used for the same domain.

Ontology Alignment Process

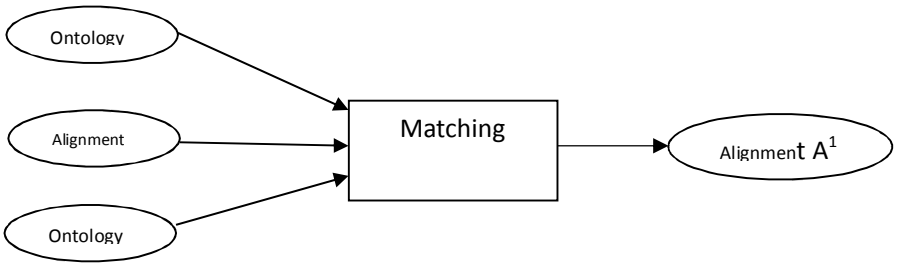


Figure 1.1 Matching Processes (Hassan, *et al*, 2012)

The figure 1.1 depicts the different phases in the generic mapping process as explained below:

Importing Ontologies: Ontologies can be specified in different languages such as RDF, RDFS and OWL, which indicates a need to convert them to a common format in order to be able to specify the alignment. Furthermore, the Ontologies need to be imported in the tool, which is used to specify the mapping.

Finding Similarities: For any two source Ontologies, the match operator returns the similarities between the Ontologies. Matching algorithms are evaluated according to the following measures precision (which is a measure of the number of correctness), recall (which is a measure of the number of completeness) and F-measure, which aggregates precision and recall. Sometimes Reference alignment which is an association of Ontologies which confirmed by domain experts is used.

First, we have to note that for simplicity we assume only two different Ontologies O_1 and O_2, which describe the same or similar domains, as input to the mapping process. The outcome of the matching process is an alignment A^1, which describes how O_1 and O_2 are related.

Given that no universal Ontology exists for the WWW, the work has focused on finding semantic correspondences between similar elements of different Ontologies, i.e.,

Ontology Alignment. Ontology Alignment can be done either by hand or using automated tools (Moa, 2008). Manual mapping becomes impractical as the size and complexity of Ontologies increases. Fully or semi automated mapping approaches include analyzing linguistic information of elements in Ontologies, treating Ontologies as structural graphs and use techniques from different fields such as statistics and data analysis, machine learning, automated reasoning, linguistics and applying heuristic rules (Moa, 2008).

A good measure of similarity is crucial to the different methods of integration of Ontologies that arise around the context of the Semantic Web. Many different matching solutions have been proposed such as TAXOMAP, ASMOV, AROMA, QOM, Anchor-flood and Falcon-AO. They take advantage of various properties of Ontologies, e.g. structures, data instances, semantics, or labels but yet have some drawbacks.

1.2 Motivation and Problem Statement

The goal of Ontology is to achieve a common and shared knowledge that can be transmitted between people and between application systems. Ontologies play an important role in achieving interoperability across Semantic Web and organization Ontology is used to enable interoperation between Web applications from different areas or from different views on one area.

In general, different Ontology sources use different data formats and modeling languages such as RDF, RDFS and OWL to represent their data and Meta data. Sources may use the same data format but differ in their structure and in the semantics of the terminology they use. Such heterogeneity is a result of the independent nature of the Ontologies and the fact that different people with different objectives in mind create these Ontologies differently.

If different people build the Ontologies about the same subject, there will be different types of Ontologies, because of many external factors. Evidently, the Ontologies share similar parts, but it is hard to be exactly the same. Currently, there are three main reasons behind the building of heterogeneous Ontologies about the same subject. The first reason is the difference in personal view among Ontology engineers. It is the primary factor, and reflects the academic background and social environment of each designer. Second, there is no standard method for building Ontologies. The theory of Ontologies is very old, but its application to computer science is not. So, it is very far from the goal of the semantic Web using the existing method directly when the Web Ontologies are built. Third, there are a variety of Ontology tools. The importance of the Ontology can be determined by benchmarking the Ontology tools. However, the tools don't use the same Ontology languages and the same methodology for building Ontologies. So, the interoperability between Ontologies is very low. In this thesis, our idea focuses on the first factor. That is the difference of personal view among Ontology engineers. Though standardization plays an important role for syntactic heterogeneity, it does not overcome structural heterogeneity which occurs as a result of the way information is structured even in homogeneous syntactic environments (Moa, 2008). For that reason, it is necessary to establish mappings among concepts of different Ontologies to capture the semantic correspondence between them. However, establishing such a correspondence is not an easy task.

Various approaches used by the current Ontology matching systems include structural technique which finds similarities among entities in structural graphs of two Ontologies, string which maps elements by analyzing entities in isolation, ignoring their relationships with other entities (Wang, *et al*, 2010). Only few have considered the use of external background knowledge as a way to obtain semantic mappings between

syntactically dissimilar Ontologies to overcome the aforementioned limitations (Marta, *et al*, 2006).

Two limitations of current Ontology mapping approaches that only rely on label and structure similarity are:-

Syntactic Approaches Do Not Provide Semantic Mappings:

Most systems combine a range of non-semantic techniques, such as string approaches (exploiting string similarity between labels) and structural approaches (relying on the structure of the mapped Ontologies). Not all systems rely on semantic techniques thus exploiting the semantics both of the mapped Ontologies.

Syntactic Approaches Fail on Dissimilar Ontologies:

Syntactic approaches fail when there is little lexical overlap between the labels of the Ontology entities, or when the Ontologies have weak or dissimilar structures. In fact, the above listed techniques are based on the hypothesis of equivalence between some forms of syntactic correspondences and semantic relations. While it is true that, in many cases, string and structural similarities can imply meaningful mappings, this hypothesis is far from being always verified. For instance, the relation between the concepts Banquet and Dinner may not be discovered on the basis of syntactical considerations, but becomes obvious when considering the meaning of these concepts (their semantics). By ignoring such semantics, syntactic techniques fail to identify several important mappings.

In Falcon-AO they have used many new concepts like Partition Based Block Matching of large scale Ontologies, Virtual Document, Similarity combination strategies and coordination rule (Hussain and Srivatsa, 2012).But still Falcon-AO does not align Ontologies with similar meanings when the structures of the two Ontologies to be aligned are different.

For example, one source might model Article but only classify them into a few categories; while the other source might make very fine-grained distinctions between types of articles based on their author, Date of publication, content and field of study etc.

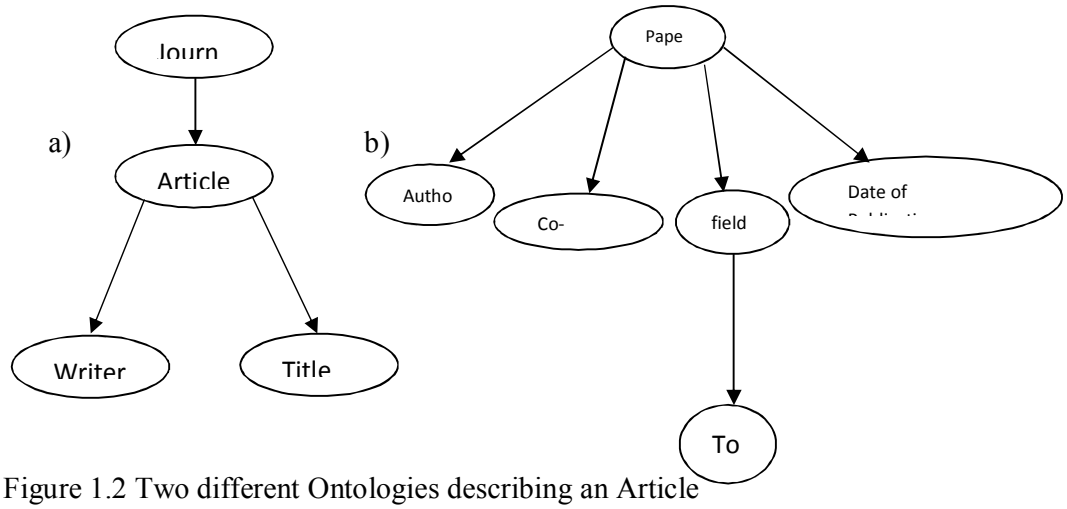


Figure 1.2 Two different Ontologies describing an Article

Though solutions to syntactic and structural heterogeneity have been developed, the problem of semantic heterogeneity is still only partially solved. Semantic heterogeneity occurs whenever two contexts do not share the same interpretation of information (e.g. Synonyms and Homonyms).

Table1.1 Semantic Relations in WordNet (Lin and Sandkuhl, 2008)

Semantic Relation	Syntactic Category	Examples
Synonym(Similar)	Noun	Pipe, Tube
	Verb	Rise, Ascend
	Adjectives	Sad, Unhappy
	Adverb	Rapidly, speedily

Using lexicon in the matching process allows for the detection of false positive

mappings and to enrich matching results by adding new mappings through discovering new results. Therefore, it is necessary to use lexicons to match Ontologies. Due to all these problems the study seeks to look at an extending an Ontology Alignment System with a lexicon database and compare the original system and the extended based on time and precision.

1.3 Research Questions

i. Can WordNet be integrated into Falcon-AO?

ii. Can Falcon-AO's precision be enhanced using WordNet with acceptable execution time penalty?

1.4 Aims and Objectives

i. The aim of this thesis is to extend Falcon-AO with WordNet in order to enhance its alignment precision.

ii. Implement the extended Falcon-AO.

iii. Evaluate original Falcon-AO and the extended Falcon-AO (precision and time).

1.5 Goal

The ultimate goal of our research is to extend Falcon-AO technique to use a lexicon database i.e. WordNet so as to utilize the meaning behind the concept names to ascertain the similarity of the concepts to match Ontologies. Therefore, enhancing the alignment results.

1.6 Methodology

The following are the steps that are set out to achieve the objectives:

i. Review literature to understand to understand how Falcon-AO works.

ii. Review literature to understand how WordNet can be integrated to Ontology Alignment systems.

iii. Study Jena API to see how rules can be applied to Ontologies.

iv. Study Java Application for WordNet to see how it can be applied to the system.

v. Extend Falcon-AO with WordNet.

vi. Implement the extended Falcon-AO system.

vii. Conduct comparative analysis of the original Falcon-AO and the Extended Falcon-AO system.

1.7 Contribution to Knowledge

This thesis made the following contribution to knowledge in the area of Ontology alignment:

i. Extended the architecture of a popular Ontology alignment system, Falcon-AO.

ii. Incorporated WordNet into the alignment process of Falcon-AO.

iii. Implemented the extended Falcon-AO

iv. Validated the extended Falcon-AO system which compares formally with the original Falcon-AO system.

1.8 Thesis Organization

The rest of this thesis is organized in the following manner. Chapter 2 elaborates on the definition of Ontology, it importance, structure and main applications of Ontology as well as a brief survey of Ontology languages which are used to express Ontology over the Web. It introduced the basic concepts of the Semantic Web and introduced relevant terms to provide a basic understanding of Ontologies. Chapter 3 discussed the methodology of the thesis i.e. the steps used to achieve our aims and objectives. Chapter 4 thoroughly described the Falcon-AO and evaluates its performance using Existing Ontology While Chapter 5 is a summary of the thesis, delimitation of the study and the outlook for future work.

CHAPTER TWO

LITERATURE REVIEW

This chapter defines Ontology, its importance, structure and some of its main applicationss as well as a brief survey of Ontology languages which are used to express Ontology over the Web. It introduces the basic concepts of the Semantic Web and all the relevant terms are explained to provide a basic understanding of Ontologies, which are the basis of this work.

2.1 ONTOLOGY

In the eighties, Ontology was introduced in artificial intelligence field to describe the concepts in real world and given new definition. There into, the most famous definition is from Gruber that Ontology is *an explicit and formal specification of a shared conceptualization* (Gruber and Thomas, 1993). Ontology is abbreviated as FESC (Formal, Explicit, and Specification of Shared Conceptualization) (Jain and Singh, 2013).where the foundations of this definition are:

Formal:-It specifies that it should be machine understandable

Explicit:-It defines type of constraints used in model.

Shared:- It means that Ontology is shared by group. It is not restricted to individuals.

Conceptualization: -It refers model of some phenomenon to identify relevant concepts of that phenomenon.

Ontologies formally express knowledge as a set of concepts and their relationships in certain area. Ontologies, which are used in order to support interoperability and common understanding between different parties, are a key component in solving the problem of semantic heterogeneity, thus enabling semantic interoperability between

different Web applications and services.

The main goal of Ontology is to achieve a common and shared knowledge that can be transmitted between people and between application systems.

The main reasons for developing Ontologies as described in (Noy and McGuinness, 2001):-

i. To share common understanding of the structure of information among people or software agents.

ii. To enable reuse of domain knowledge and to introduce standards.

iii. To make domain assumptions explicit.

iv. To separate domain knowledge from the operational knowledge and finally to analyze domain knowledge.

2.1.1 Ontology components

Ontologies share many structural similarities, regardless of the language in which they are expressed. Most Ontology describes individuals (instances), classes (concepts), attributes, and relations. Common components of Ontologies include as described in (Kaushal, 2011).

i. **A Concept: -** set or collection of objects also known as classes, are used in a broad sense. Concepts, also called Classes**,** Types or Universals are a core component of most Ontologies. A Concept represents a group of different Individuals that share common characteristics, which may be more or less specific. For example, humans share certain characteristics, such as a set of specific body parts, the ability to speak a complex language.

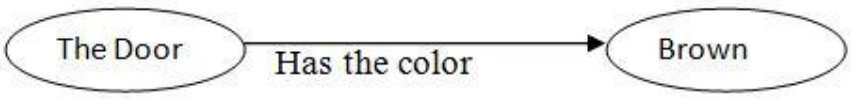
ii. **An Instance**: - (also known as an individual) is the “ground-level” component of an Ontology which represents a specific object or element of a concept or class. The individuals in Ontology may include concrete objects such as people, animals, tables, automobiles, molecules, and planets.

iii. **A Relation**: - These are ways in which classes and individuals can be related to one another. More specifically, it describes the relationship between the first concept, represented in the domain, and the second, represented in the range. For example, “dancing” could be represented as a relationship between the concept “person” (which is a concept in the domain) and “club” (which is a concept in the range).

2.1.2 Ontology languages

i. Resource description framework (RDF)

RDF is a data model. It is based upon the idea of making statements about resources (in particular Web resources) in the form of subject-predicate-object expressions. These expressions are known as *triples* in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. For example a way to represent the notion "The door has the color brown" in RDF is as the triple: a subject denoting "the door", a predicate denoting "has the color", and an object denoting "brown".



RDF has a simple data model that is easy for applications to process and manipulate (Klyne and Carroll, 2003).RDF is capable of representing data on and exchanging knowledge over the Web. It was developed to be understood by computers,

facilitating interoperability between applications. In other words, it is a framework for using and representing metadata and describing the semantics of information about Web resources in a way that is accessible to machines. RDF, which is recommended by the W3C, it is based upon XML, which is designed for syntax, while RDF is intended for semantics. As has been mentioned, it is a framework for describing Web resources, which is why it has become a common method of describing the properties, time, information and content of Web resources, so that it can be read and understood by computer applications.

ii. Resource Description Framework Schema (RDFS)

RDFS is Schema Language for RDF. It has been built upon the XML and RDF models and upon syntax. It allows specifying a number of useful constraints on the individuals and relationships used in RDF triples. It allows declaring objects and subjects as instances of certain classes .It provides a machine-understandable system for defining the vocabularies needed for such applications or descriptive vocabularies. In other words, it is a group of RDF resources that can be used to define or express the properties of other RDF resources which define application specific RDF vocabularies. At the same time, RDF(S) helps developers to describe classes and properties in a specific way and to specify relationships between those properties and classes, allowing combinations between classes, properties or values. In other words, RDFS is used to define RDF vocabularies.

iii. Web Ontology Language (OWL)

OWL (Web Ontology Language) extends RDFS with the possibility to express additional constraints. Like RDFS, OWL statements can be expressed as RDF triplets using some specific predicates and objects used as OWL keywords with a

particular meaning.

There are many constructs expressible in OWL that provides considerable modeling flexibility and expressiveness for the Semantic Web. This allows better understanding inferences when considering facts described with RDF triples and constraints in OWL.

OWL offers a number of rich semantic constructs, namely class disjoint, functional constraint, intentional class definition, class union and intersection, etc.

Sublanguages of OWL

OWL provides three increasingly expressive sublanguages designed for use by communities of implementers and users.

OWL Lite: - Supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. It also has a lower formal complexity than OWL DL.

OWL DL: - supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL (McGuiness and Harmelen, 2004)**.**

OWL Full: - is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an Ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full (Taye, 2010).

2.1.3 Ontology applications

Over the years, Ontology has become a popular research topic in a range of disciplines, with the aim of increasing understanding of and building an agreement in a given area of knowledge. Ontology also leads to the sharing of knowledge between systems and people.

i. **Semantic Web** (Taye, 2010): Ontology plays an important role in the Semantic Web in supporting information exchange across distributed environments. The Semantic Web gives structure to the content of Web pages and enables computers to perform sophisticated tasks for people. Inspired by this vision, researchers throughout the world have been engaged in researching Ontology and the Semantic Web.

ii. **Semantic Web Service Discovery :-(** Clerkin, *et al,* 2002**)** Most of the Web's existing content is designed to be read and understood by humans, and cannot readily be parsed and processed by software agents. The central idea behind the Semantic Web is to develop and use machine-understandable languages for the expression of the semantic content of Web pages. This promises to enhance the ability of software agents to navigate the Web's information space and carry out tasks for humans, without the need for sophisticated artificial intelligence.

iii. **Interoperability:-** Semantic interoperability is a crucial problem in networked systems where many different and independent enterprise parties need to cooperate and share heterogeneous information resources often in response to opportunities or challenges that cannot be anticipated in advance and require a rapid response. Accessing heterogeneous and distributed informational resources in a coordinated and virtual way requires appropriate semantic interoperability techniques to enable a seamless access and retrieval of the right information resources, while preserving the information representation and management requirements of each single party involved in the Alliance

iv. **Knowledge Acquisition**: - Knowledge acquisition refers to how machines imitate human to gain knowledge in artificial intelligence field and knowledge project system (Cao, *et al*, 2012). The bottleneck of knowledge acquisition has long been recognized as a major stumbling block to the widespread deployment of knowledge based systems. In order to carry out the task it must be possible to proceed in a systematic way, and to put the knowledge acquired into a coherent structure. Ontology is an important framework to drive the knowledge acquisition process. It tells you what precisely it is she/he needs to acquire knowledge about, and what knowledge needs to be acquired about these things, and what information to be ignored. The structure of the Ontology provides a structure for the acquisition process. Ontology identifies gaps in the acquired knowledge that need to be completed and helps to determine when the acquisition process is complete.

2.1.4 Ontology problems

Recently, research has been taken towards the standardization of Ontologies and Ontology languages. While it may be possible to introduce standards in language and syntax, it becomes impractical to promote the use of one consistent Ontology or

knowledge base, since it is impossible for human experts in scientific domains to reach full and complete agreement. In this case it becomes more desirable that the system can deal with inconsistencies and different Ontologies. Integration of these Ontologies is an important research area. For that reason, it is necessary to establish matching among concepts of different Ontologies to capture the semantic correspondence between them.

2.1.5 Definition

The terms mapping, matching and alignment are frequently used in the work about combining Ontologies as summarized by (Hassan, *et al*, 2012).Based on recent studies about combining Ontologies, the terms used in this report are defined as follows:

i. **Ontology merging**: - Combine two Ontologies from the same subject area into a new Ontology.
ii. **Ontology Integration: -** Combine two Ontologies from different subject areas into a new Ontology.
iii. **Ontology Mapping: -** Finding equal parts in different source Ontologies.
iv. **Ontology matching: -** Find similar parts in the source Ontologies or finding translation rules between Ontologies.
v. **Ontology Alignment**: - Identify correspondences between the source Ontologies. Ontology Alignment is the main approach to solve the heterogeneity of information between heterogeneous Ontologies in semantic Web. The objective of Ontology mapping is to extract an alignment between these two Ontologies. An alignment or Matching consists of a set of correspondences between their entities. Ontology mapping is used for several applications such as Ontology engineering, information integration, peer to peer information sharing.

2.2 SEMANTIC WEB

The current World Wide Web has over a billion pages but majority of them are in human readable format as a result software agents cannot understand and process this information. Therefore, most of the capability of the Web has so far remained unutilized.

In reaction to this problem researchers created the vision of the semantic Web which is divided into two major points.

i. To improve its usability as medium for collaboration

ii. To ensure that its content can be understood by machine.

Tim Berners-lee the inventor of the World Wide Web stated that the semantic Web is "*an extension of the current Web, in which information is given a well-defined meaning, better enabling computers and people to work in cooperation"*. The Semantic Web is distinguished by a more meaningful representation of information for humans and computers, providing a description of its contents and services in machine-readable form; moreover, it enables services to be automatically annotated, discovered, published, composed and advertised. It thereby facilitates interoperability and the sharing of knowledge over the Web. Its main goal is therefore to make information on the Web accessible and understandable by humans and computers.

The following points describes the role of Ontology in development of Semantic Web:

i. Ontology is considered as backbone of Software. Since Semantic Web translates the given data into machine understandable language using concept of Ontologies.

ii. Ontology development is a cooperative process; it allows different peoples to express their views on given domain.

iii. It gives complete description of problem that can be communicated among people and application systems.

iv. The use of Ontologies has overcome the limitations of keyword based search and led to development of Semantic Web.

2.3 ONTOLOGY MATCHING TECHNIQUES

This section reviews techniques currently used for Ontology matching. These techniques are classified into element level and structure level techniques.

2.3.1Element-level techniques

These techniques view Ontology entities or their instances as isolated from other entities or instances. They are classified into string-based, language-based, and constraints-based, described as follows:

i. String-based techniques

These techniques are used to match names of the entities in Ontologies. Such techniques are based on the similarity of the names of entities, considered strings. The more similar the strings, the more likely they denote the same concepts. There are numerous methods introduced for string similarity matching. The most frequently used methods are:

Edit Distance: in this method of matching two entities, a minimal cost of operations to be applied on one entity in order to obtain the other entity is considered.

Normalization: To improve the matching results between strings, a normalization operation is performed, usually before matching. In particular, these operations are case normalization, blank normalization and punctuation elimination.

String equality: the string equality method basically returns 0 if the input strings compared are not identical, and 1 if they are. An example of such a method is the Hamming distance.

Substring test: This identifies the ratio of common subparts between two strings. Also, it is used to compute if a string is a substring of another string. i.e., a prefix or suffix.

Token-based distances: Such a method considers a string as a set of words. These methods are used to split long strings (strings that are composed of many words) into independent tokens.

ii. Constraints-based techniques

To calculate the similarity between entities, there are different methods proposed based on constraints which compare the properties, data types, and domains of entities.

Property comparison: - When the properties of two classes are similar (similar names and types), it is more likely that these two classes are similar.

Data type comparison: - This compares the way in which the values are represented, e.g. integer, float, string.

Domain comparison: - Depending on the entities to be considered, what can be reached from a property can be different in classes, these are domains, in individuals, these are values.

2.3.2 Structure-level techniques

In contrast to element-based techniques, structure-based techniques compare the two entities from two Ontologies with regards to the relations of these entities with other entities in the Ontologies: the more similar the two entities are, the more alike their relation would be. Mainly, there are two well-known structure level techniques: graph-

based techniques and taxonomy-based techniques, described as follows:

i. **Graph-based techniques**
These techniques consider the Ontologies to be matched as labeled graphs. The basic idea here is that, if two nodes from two Ontologies are similar, their neighbors should also somehow be similar.

ii. **Taxonomy-based techniques**
These techniques are basically graph-based techniques which consider only the specialization relation. The basic idea they focus on is that an is-a relationship links terms that are already similar, therefore their neighbors may also be similar.

Matching Ontologies using their structure information is important as it allows all the relations between entities to be taken into account. The most common techniques used for Ontology matching are taxonomy-based, since taxonomies play a pivotal role in describing Ontologies.

iii. **Model-based techniques**
Model-based (semantically grounded) algorithms handle the input based on its semantic interpretation, e.g., model-theoretic semantics. The intuition is that if two entities are the same, then they share the same interpretations. Thus, they are well grounded deductive methods.

2.3.3 Use of External Resources

These techniques measure the semantic relatedness of concepts, for which they consider names as words in some natural language. They use Natural Language Processing techniques to extract meaningful terms from the text. Typically, WordNet is used either to simply find close relationships such as synonymy between the two terms, or to compute some kind of semantic distance between them in order to decide if a

relationship should hold. Usually, they are applied to words (names) of entities. The matching similarity is determined based on linguistic relations between words, such as synonyms and hyponyms.

Table 2.1 Ontology Matching Techniques Classification

Ontology Matching Technique	**Description**	**Further Classification**	**Remarks**
Syntactic	Considers Ontology entities or their instances in isolation from their relations with other entities or their instances.	String Base Edit Distance Normalization String test String Equality Token Based Distance Language Base Constraint Base Property Comparison Data Type Comparison Domain Comparison	Helpful where the matching comparison of the Ontology is based on text strings or on the language.
Structure-level	Considers the Ontology entities or their instances to compare their relations with other entities or their instances	Graph Base Taxonomy Base Model Base	Valuable technique for evaluating and aligning internal structure of the entities
Use of External Resources	use of external resources such as dictionaries and lexicons this methods essentially match terms which are semantically related, using an external lexicon or dictionary	-	Helpful if the information of entities compared is limited and needed to compare additional data or support other matching techniques for detecting where frequently erroneous or misleading correspondences occur

2.4 REVIEW OF ONTOLOGY MATCHING SYSTEMS

The systems reviewed are based on the input ontologies used, one or a combination of the approaches described in 2.3, output and also the limitations of some of the systems.

2.4.1 Anchor-Flood

The Anchor-Flood was developed by (MH & Masaki, 2009) in Toyohashi University of Technology, Aichi, Japan. The approach aims at handling efficiently particularly large Ontologies. It inputs Ontologies in RDFS and OWL and outputs 1:1 alignments. The system starts with a pair of similar concepts from two Ontologies called an anchor, e.g.,

all exactly matched normalized concepts are considered as anchors. Then, it gradually proceeds by analyzing the neighbors, i.e., super-concepts, sub-concepts, siblings, of each anchor, thereby building small segments (fragments) out of the Ontologies to be matched. The size of the segments is defined dynamically starting from an anchor and exploring the neighboring concepts until either all the collected concepts are explored or no new matching pairs are found. The system focuses on (local) segment-to-segment comparisons, thus it does not consider the entire Ontologies which improves the system scalability. It outputs a set of correspondences between concepts and properties of the semantically connected segments.

For determining the correspondences between segments the approach relies on string, structural similarity measures, which are further aggregated by also considering probable misalignments.

The similarity between two concepts is determined by the ratio of the number of terminologically similar direct super-concepts on the number of total direct super-concepts. Retrieved (local) matching pairs are considered as anchors for further processing. The process is repeated until there are no more matching pairs to be processed.

The drawbacks of this method are: It Ignores certain distantly placed aligned pairs since segments are constructed from neighbor of anchors. Semantic similarity between entities is not explored. Only exact string match of concepts, properties and instances are considered as anchors which lead to inefficient alignments (Saruladha, *et al*, 2011).

2.4.2 **AROMA** (Association Rule Ontology Matching Approach**)**

AROMA aims at finding subsumption relations (and also equivalence) between entities (classes or properties) of two distinct hierarchical structures provided with textual data. The approach relies on the following assumption:

An entity A will be more specific than or equivalent to an entity B if the vocabulary (i.e. terms and also data) used to describe A, its descendants, and its instances tend to be included in that of B.

AROMA is divided into two successive main parts: The first is term or data sets extraction and the second is discovery of subsumption relations. The first part constructs a set of relevant terms and/or data values for each entity. This first processing task depends on the type of taxonomy considered (textual hierarchy or OWL Ontology). For OWL Ontologies, we extract the vocabulary of entities from their annotations and individual values. The second part of AROMA discovers the subsumption

AROMA is a time efficient matcher compared to other matching alignments. AROMA takes very less memory space in order to find alignments. But as stated in (Hussain and Srivatsa, 2012) in some cases the precision is also degraded due to the subsumption correspondences it returns**.**

2.4.3 QOM (Quick Ontology mapping)

In (Ehrig and Staab, 2004) the authors present an Ontology Mapping Method, QOM (Quick Ontology Mapping), which emphasizes speed over accuracy of results, which is different than previous research that was focused on getting the best results possible (in terms of accuracy). The author's argument is that the loss in accuracy is quite marginal, yet the gain in efficiency is tremendous that makes QOM a much more feasible method for real-world practical applications where often the Ontology schemas can get quite large.

QOM extends Native Ontology Mapping (NOM), and the basic high-level steps of the algorithm proceed as follows:

i. Feature Engineering-The Ontologies must be in a format equivalent to the RDFS format.

ii. Search Step Selection - Heuristics are used to lower the number of candidate mappings; this is done to reduce the run-time complexity and therefore improve the performance, since this reduces the number of candidate mapping pairs that have to be compared. First are the candidate mappings that must be investigated. Second, mappings are ordered and some of them are rejected to gain efficiency.

iii. Similarly Computation – QOM is based on a wide range of Ontology feature and heuristic combinations. To keep up the high quality of mapping results the retained using as many Ontology features as possible. But, in order to optimize QOM, a complete pair-wise comparison of trees and instead opted for a partially complete top-down strategy. With this strategy, the authors only retrieve partially filled sets of data, consciously reducing accuracy for performance.

iv. Similarity Aggregation –There is emphasis on high individual similarities and no emphasis on low individual similarities. This is done by putting first a weight on individual similarity results with a sigmoid function and then summing the modified values.

v. Interpretation – Similarity is interpreted in two ways. First, a threshold is applied to reject any cases that are not that similar. Secondly, objectivity relationships in the mappings are maintained such that any similarity that violates this relationship is discarded.

vi. Iteration - QOM iterates to find mappings based on knowledge structures.

2.4.4 TaxoMap Matching Technique

TaxoMap (Hamdi *et al*,2009) is an Ontology matching algorithm designed to align two OWL Ontologies consisting of two partitioning algorithm namely Anchor Partition Parti ntion (APP) and Partition Anchor Partition (PAP) which have been designed to take the alignment objective into account in the partitioning process. The most structured Ontology is referred as target Ontology and the less structured is referred as source Ontology. PAP is suitable for structured vs. unstructured Ontology matching and APP is suitable for structured vs. structured Ontology matching. The entity pair one from each of the two Ontologies which has identical labels is called as anchors which will be used in both PAP and APP. The alignment is based on string and structure (subclass) similarity measure.

Anchor Partition Partition (APP):

i. Identify the set of anchors across Ontologies.
ii. Partition both the target Ontology and source Ontology by modifying PBM matcher in order to take into account shared anchors.
iii. Align blocks that share maximal number of anchors.

Partition Anchor Partition (PAP):

i. Use PBM matcher to partition the target Ontology into set of blocks.
ii. Identify the set of anchors between two Ontologies. This set will be the center of the future block which will be generated from the source Ontology.
iii. Use PBM matcher to partition the source Ontology around the identified centers.
iv. Align each block with the corresponding block.

The drawbacks of this method are as follows from (Saruladha, *et al*, 2011): The effectiveness of this method depends on the availability of identical labels across

Ontologies. Only labels and hierarchy structure is used for matching and hence comparatively less recall.

2.4.5 ASMOV (Automated Semantic Matching of Ontologies with Verification)

ASMOV (Jean-Mary and Kabuka, 2009) is a good rule that uses lexical and structural characteristics of two Ontologies to repetitive calculate a similarity measure between them, derives an alignment, and then verifies it to ensure that it does not contain semantic inconsistencies (Euzenat,*et al*, 2013). It is an automated Ontology mapping tool that iteratively calculates the similarity between concepts in Ontologies by analyzing four features, i.e., textual description (id, label, and comment), external structure (parents and children), internal structure (property restrictions for classes; types, domains, and ranges for properties), and individual similarity. It combines a comprehensive set of element-level and structure-level measures of similarity with a technique that uses formal semantics to verify whether computed correspondences comply with desired characteristics. The ASMOV process is an iterative process divided into two main components:

i. Similarity calculation
ii. Semantic verification.

ASMOV receives as input two Ontologies in OWL-DL format to be matched, and an optional input alignment, containing a set of predetermined correspondences. The first stage is the similarity calculation process which computes a similarity value between all possible pairs of entities, one from each of the two Ontologies, and uses the optional input alignment to adjust any calculated measures. ASMOV still needs to improve its ability to work with very large Ontologies and resources, need to re-examine the use of an appropriate threshold value to optimize accuracy (Hussain and Srivatsa, 2012).When the given two Ontologies are dissimilar effectiveness is decreased. Sometime semantic

verification system eliminates too many alignments leading to less recall. For each iteration the ASMOV needs polynomial time, thus leading to inefficiency (Saruladha, *et al*, 2011).

2.4.6 FALCON-AO

Falcon-AO is an Ontology Alignment System. It is an infrastructure for Semantic Web applications, which aims at providing fundamental technology for finding, aligning and learning Ontologies, and ultimately for capturing knowledge from the Web via an Ontology-driven approach. Falcon-AO was developed by Department of Computer Science and engineering South East University, China. Falcon-AO is implemented in Java, and presently, it is an open source project under the Apache 2.0 license. There are four matchers integrated in Falcon-AO: V-Doc a linguistic matcher, GMO an iterative structural matcher and PBM partition-based Block matcher, I-Sub a String Matcher, as stated in (Hu , 2010)**.**

Draw backs of Falcon -AO

i. While expressing the same thing, people might use synonyms.

   Falcon-AO cannot provide alignments with semantic relationships.

ii. Falcon-AO uses only String and Structural similarity.

Advantages of Falcon–AO

According to (Hussain and Srivatsa, 2012) Falcon-AO as analyzed against other system

i. Is practical and popular choice for matching Web Ontologies expressed in RDF(S) and OWL

ii. It is a free and open source software

iii. Can fulfill various matching tasks especially matching large Ontologies

iv. Is time Economic: All the tasks can be carried out in a sensible time even on an ordinary personal computer. For small Ontologies, Falcon-AO can complete the matching process within several seconds, even for large-scale Ontologies, Falcon-AO can accomplish them within a few hours.

Table 2.2 Comparison of some Matching Systems

System	Input	Output Alignment	GUI	Alignment Techniques	Limitation
Anchor-Flood	RDFS,OWL	1:1	No	String and Structural	Reduced search space to achieve better efficiency
AROMA	OWL	1:n	yes	String and structural	Restricted to OWL-Lite No inference is performed on the Ontology Only descriptive knowledge is taken into account
QOM	RDFS/OWL	1:1		String and the use of external resource: WordNet	It emphasizes speed over accuracy of results.
TAXOMAP	OWL	1:1	yes	String and Structural,	The effectiveness of this method depends on the availability of identical labels across Ontologies. Only labels and hierarchy structure is used for matching and hence comparatively less recall.
ASMOV	OWL	n:m	Yes	String and structural, Use of external resources : WordNet and UMLS	When the given two Ontologies are dissimilar effectiveness is decreased. Semantic verification system eliminates too many alignments leading to less recall ASMOV needs polynomial time for each Iteration which leads to inefficiency.
Falcon-AO	RDFS,OWL	1:1	yes	String , Structural and PBM	When the structure of the given two Ontologies are dissimilar it cannot detect mappings Uses only string and structural techniques

CHAPTER THREE

DESIGN OF EXTENDED FALCON-AO

This Chapter discusses the methodology of the thesis that is the steps taken to achieve our aims and objectives. The chapter focuses on Falcon-AO System which represents the application context of the methods proposed in this thesis.

The rest of the Chapter is organized as follows: section 3.1describes the system requirements, Section 3.2 describes the Extended Falcon-AO System, its Architecture, and Components while in Section 3.3 other components of the system is described.

3.1 SYSTEM REQUIREMENT

i. Netbeans IDE**:-** is an Integrated Development Environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5.It was used to implement the processes in the application.

ii. WordNet**: -** is a Lexical Database for the English language. It groups English words into sets of synonyms called *synsets*, WordNet groups words together based on their meanings.

iii. JENA API**:** Jena API is a Java frame work for building Semantic Web Application, it provides extensive Java libraries that help developers to develop code that handles RDF, RDFS and SPARQL i.e. it helps to read and write to Ontology file and also implement Query language. It includes a rule based Inference engine to perform reasoning based on OWL and RDFS Ontologies and a variety of storage strategies to store RDF triples in memory or on disk.

Table 3.1 Some Classes and Interfaces of Jena API with their Descriptions

Classes and Interfaces	**Description**
OntDocumentManager	Is a class used for managing Ontology document, including loading imported documents and locally catching document to improve load performance
SetProcessImports	Sets the policy flag for processing imports of loaded Ontologies
OntModelSpec	Is a class that captures all description of the components of an Ontology model, including the storage scheme, reasoned and language profile.
Model Factory	Is a class used to create different kinds of models
CreateOntologyModel	Is a method in the modelFactory class creates a new Ontology model which will process in-memory models of Ontologies expressed the default Ontology language (OWL).
Model	Is an interface that is used to create resources, properties and statements. It also helps to read and write to an existing model or dataset.
OntModelSpec	This is an interface that is a variant of the Model class. But it is used to provide extra capabilities for handling Ontologies.

iv. JAWS**: -** Java API for WordNet searching is an API that provides Java application with the ability to retrieve data from the WordNet database. It is a simple and fast API that is compatible with 2.1 and 3.0 version of the WordNet database files. It can be used with Java 1.4 and later. To use JAWS in your application you must do the following.

i. Download and install a copy a full version of WordNet database files.

ii. Download and install the Java archive (JAR) files containing the compiled JAWS code.

3.2 FALCON-AO

Falcon-AO is the base system for this thesis.

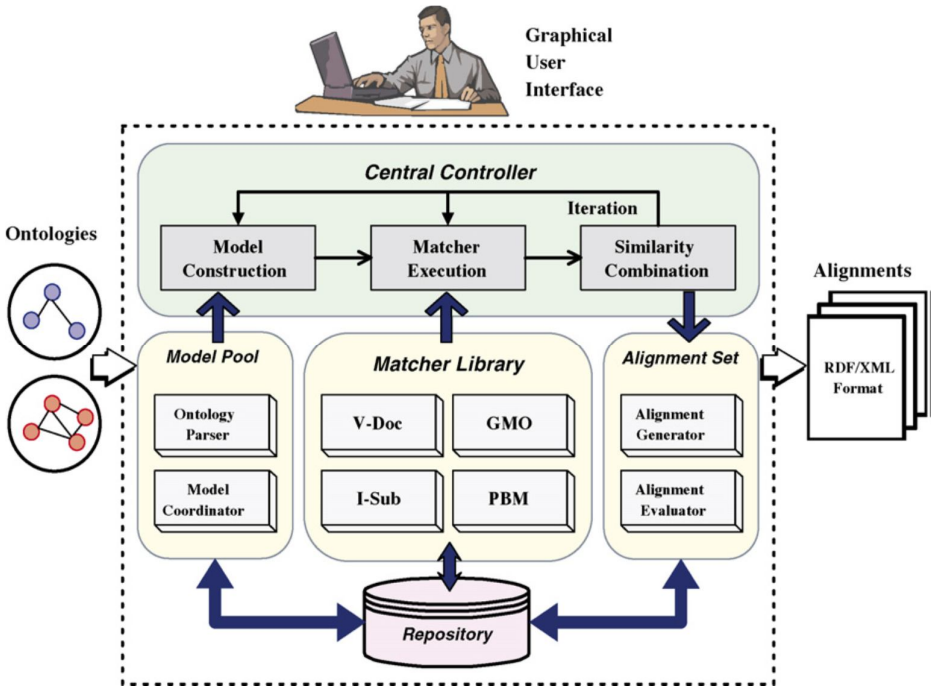


Figure 3.1 Architecture of Falcon-AO (Hu, 2010)

It has five major components Model pool, Matcher Library, Alignment Set, Central controller and Repository.

3.3 EXTENDED FALCON-AO

In our system, the extended Falcon-AO, Falcon-AO which is the base system was extended with an external resource WordNet to search for synonymous entities in the input ontologies. Our extended architecture is represented in Figure 3.1.

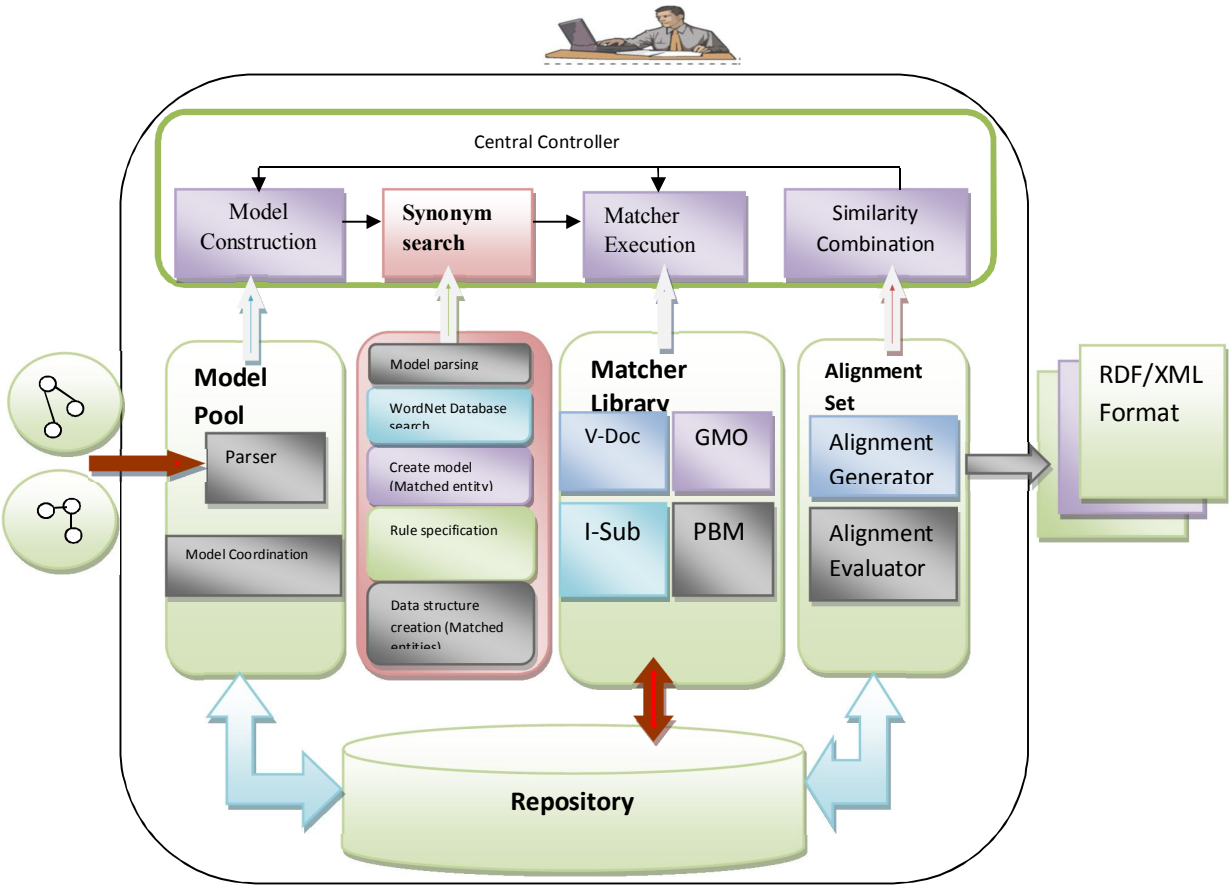


Figure 3.2 Architecture of the Extended Falcon-AO

3.3.1 Synonym Search Component

The synonym search component finds synonyms of each entity or instance of the first Ontology and tries to match the entities in the second ontology. It has five sub components which are:

i. Model Parsing and Entity Retrieval
ii. WordNet Database Search
iii. Model Creation
iv. Rule Specification
v. Data structure Creation

i. **Model parsing and entity retrieval: -** This is the first process of the component to align ontologies. In aligning ontologies there is need to access information from different ontologies with different structure/format. To do this, the system validates the two input ontologies and also create an internal representation of the ontologies by transforming each term to a standard form that can be easily recognized, then retrieves all the entities in the ontologies. In this work all these are done using Jena API which is a Java framework for building semantic web applications.

ii. **WordNet search: -** A synonym set or synsets is a set of words for a definition of a term. A word has a synsetsfor each of its definition. After retrieving all the classes in the two ontologies from model parsing and entity retrieval, it then iteratively walks through the entities of the first ontology to find synonyms of each of those entities. For each of the synonyms of the entities in the first ontology, a corresponding matched entity is searched for in the second ontology. Whenever there is a match, the two entities from the two ontologies are marked as satisfying the requirement for matching. The words are determined to be semantically equal or equivalent. In the WordNet search JAWS (Java Application for WordNet Search) which is a fast and simple API was used in theJava applications built on NetBeans integrated development environment to retrieve data from the WordNet database.

iii. **Create model:** Once two entities from the input ontologies are found to be equal in meaning a temporary empty ontology model is created which will store all the classes and instances found to be similar in meaning from WordNet database. When working with ontologies in Jena, all of the information remains encoded as RDF triples stored in the RDF model. The ontology application programming interface doesn't change the RDF representation of ontologies. What it does is add a set of convenience classes and methods that make it easier for you to manipulate the RDF triples.

iv. **Rule specification: -**Once two entities from the two different ontologies have been found to match by the WordNet as synonyms and then temporarily stored in the temporary ontology model, the Rule specification stage creates and applies the **SameAs** rule provided by Jena. Where the **addSameAs** method was used to indicate that two given entities should be modelled as equivalent in the temporary ontology model. For any entity with a matching term it specifies that the entity from the first ontology is equivalent to that of the second ontology

v. **Data structure: -** After all the processes have been carried out and the *sameAs* specified for similar entities from both ontologies, these similar entities that have been modelled in the temporary ontology are then retrieved as paired equivalent entities which are then stored in a data structure (similar to that of the original Falcon) to be further merged with what Falcon–AO has been able to match using the four matchers. Therefore, the WordNet matched entities were organized so that they could be easily provided to users. All entities matched by WordNet are then joined with entities that the other matchers were able to match and then displayed as result.

3.3.2 Extended Falcon-AO System Flow

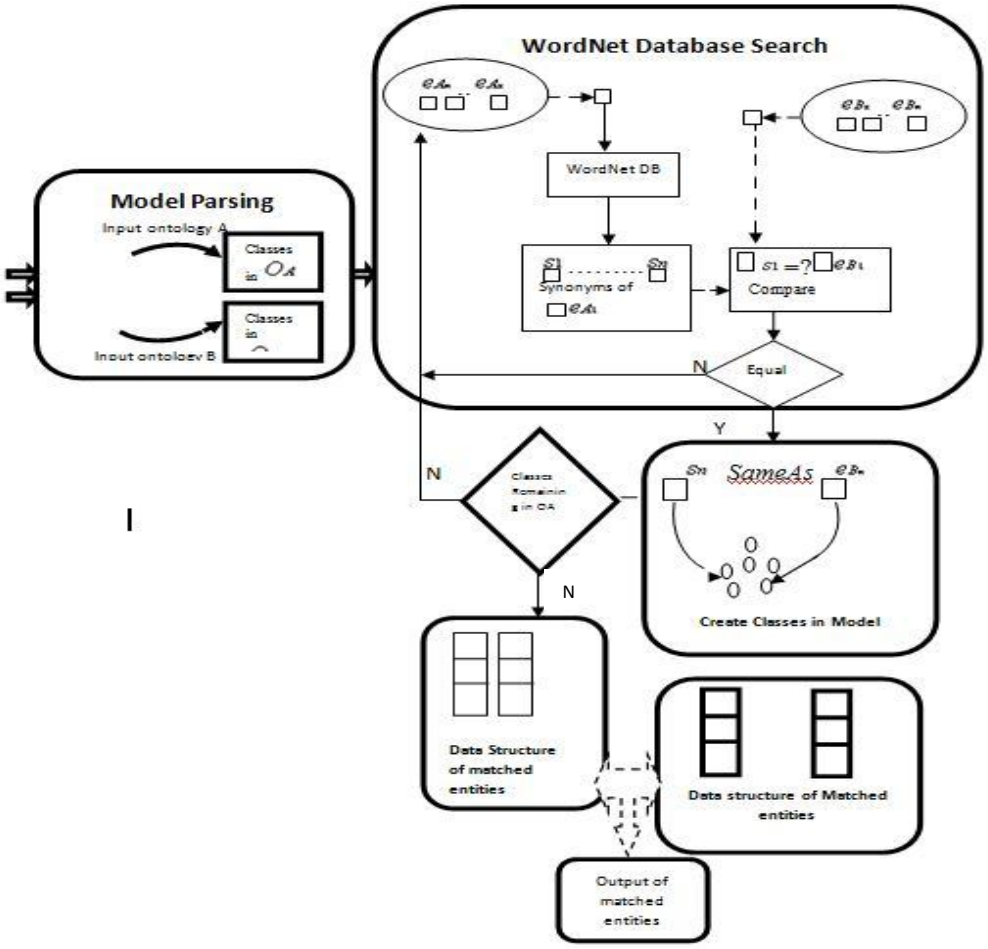


Figure 3.3 System flow of Extended Falcon-AO

The system flow shown in Fig 3.2 simply explains the synonym search added component in brief. The model parsing and entity retrieval transforms the input ontologies as required and then removes all the entities ,then passes the classes and instances of the first Ontology to the WordNet database to search for their synonyms. It then searches through the second Ontology for a match, if a match is found they are saved in into a temporary ontology. A SameAS rule is then applied to the entities, then these entities are stored into a data structure similar to that of the original Falcon-AO.

3.4 OTHER COMPONENTS OF THE EXTENDED FALCON-AO

The extended Falcon-AO System consists of five more Components which include Model Pool, Matcher library, Alignment Set, Central Controller, Repository. The sample ontologies in Figure 3.3 are going to be used in explaining the matchers in the extended Falcon-AO.

3.4.1 Model pool

The model pool contains the Ontology parser and the model coordination. Model Pool parses input Ontologies in memory models by Jena and adjusts models by using a set of Coordination rules while the model coordinator Implements coordination rules to remove inessential Axioms (statements) and also decreases the structural heterogeneity between the Ontologies to be matched. There are three categories of coordination rules that can be assigned to elementary matcher According to (Jian, et al , 2005) this include:

i. Removing redundant statements
ii. Inferring omitted statements
iii. Reconstructing list structures

In Falcon-AO Ontology parsing and model coordination are done by Jena.

3.4.2 Matcher library

The matcher library contains the V-doc and I-sub which are two Linguistic matchers, GMO a structural matcher and PBM partition- based block Matcher for Large Ontologies.

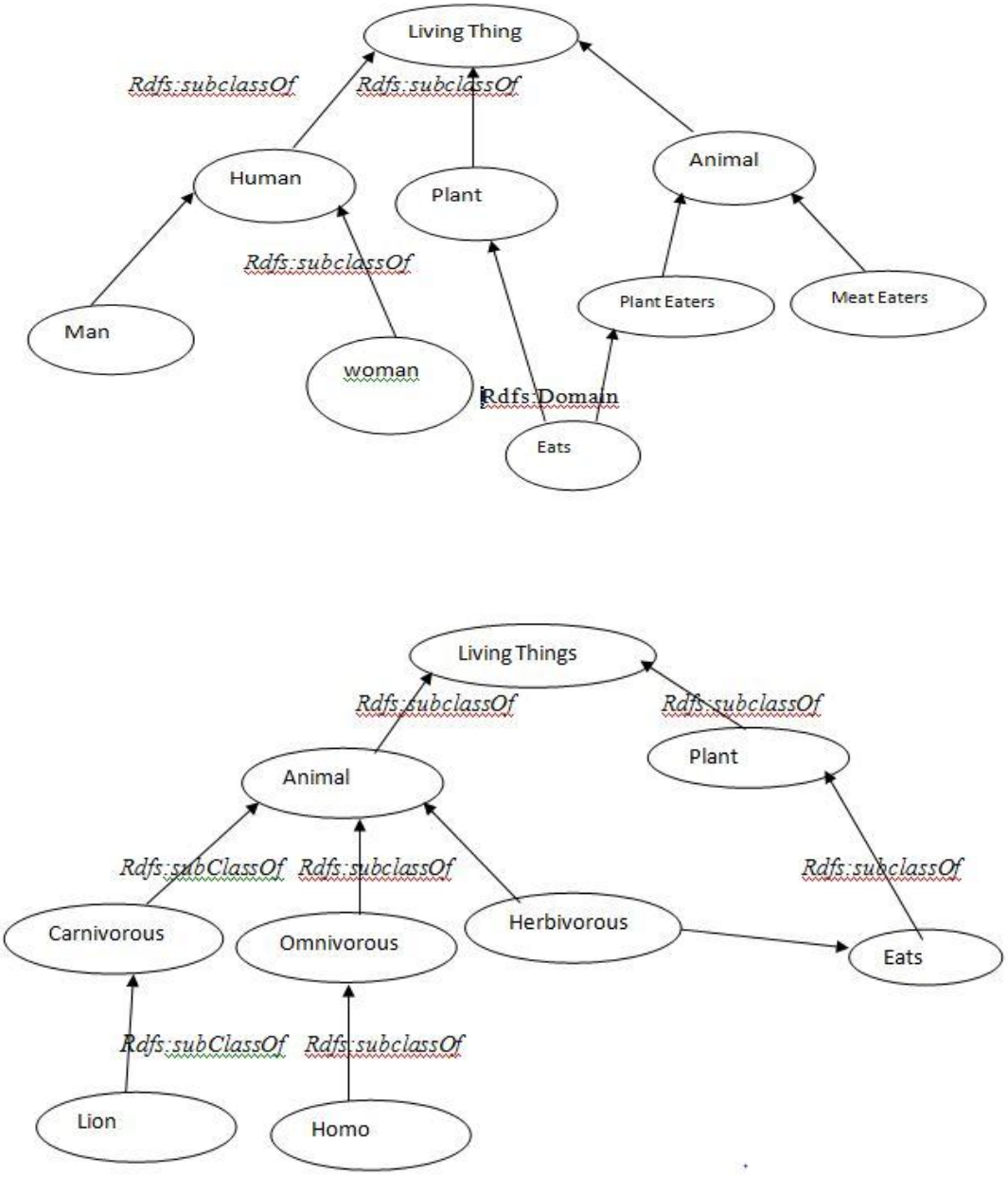


Figure 3.4 Ontologies of the same domain with different structures

i. I-SUB

I-Sub is a linguistic matcher simply based on the string comparison techniques. Its uniqueness is not only in the similarities between the descriptions of domain entities are calculated but also their differences are examined.

ii. PBM (PARTITION-BASED BLOCK MATCHER)

Is a partition-based block Matcher for large Ontologies. It uses a divide and conquers approach to find Block mappings between large scale Ontologies. Normally, large Ontologies pose a significant challenge to the present day Ontology matching systems. The divide and conquer approach provides two advantages to the system.

1. It Avoids system suffering from lack of memory

2. It decreases execution time

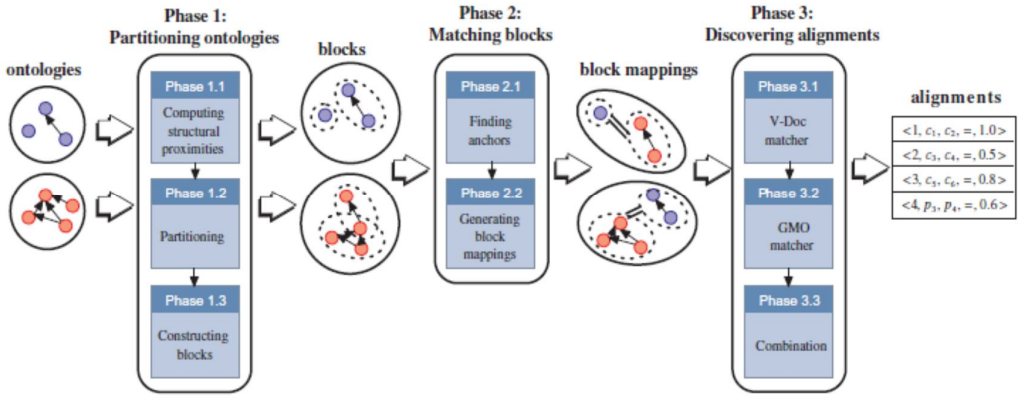


Figure 3.5 Overview of PBM approach (Hu, *et al,* 2010)

The framework shows the input of two large Ontologies to be matched.

PHASE 1

Parsing: In this stage both Ontologies are parsed and all entities retrieved.

Entities = $\{e_1, e_2 \ldots e_n\}$

Then the entities are classified into Classes and Properties.

$C = \{c_1, c_2, c_3 \ldots c_n\}$

$P = \{p_1, p_2, p_3 \ \ldots p_n\}$

Computing Structural Proximities:-Structural Proximities between classes are defined based on how closely they are related in the hierarchies of rdfs: subClassOf relationship

Equation

$$\text{Prox}\ (C_i, C_j) = \frac{2.depth(C_{ij})}{depth(C_i) + depth(C_j)}$$

(Hu, *et al* , 2008) (3.1)

Where C_{ij} is the common superclass of C_i and C_j, and depth is the level of original class in the hierarchy.

Example: Using the sample ontology$_1$ from figure 3.3 to show the structural proximity between the entities. Taking the depth of living thing as 1, Human, plant and Animal as 2 while Man, Woman, Herbivores and lion as 3

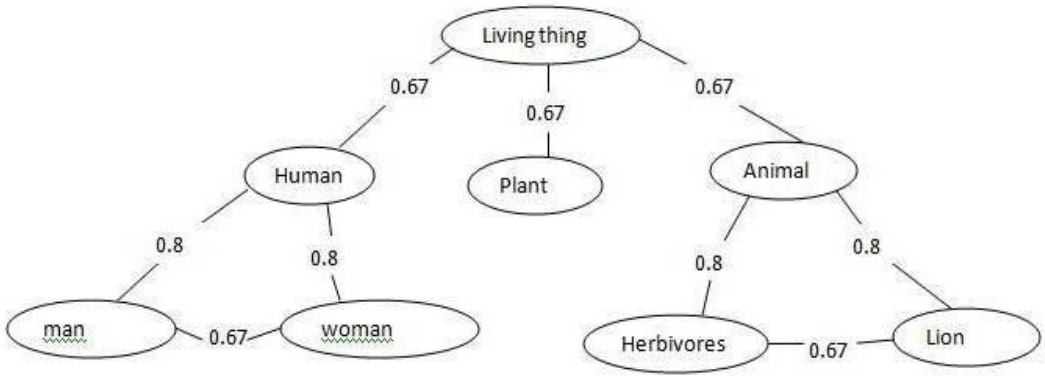


Figure 3.6 Sample Ontology 1with values of structural proximity

Prox (Animal, Herbivores) = $\frac{2.2}{2+3} = \frac{4}{5} = 0.8$

Partitioning: - It first divides the entities of each Ontology into groups of clusters using the structural proximity. A cluster here is a set of similar entities grouped together. It

then builds blocks. Example of a cluster from the sample Ontology above can be seen in Figure 3.7 .

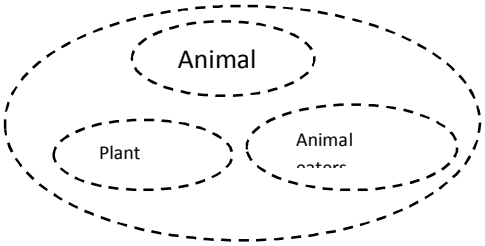


Figure 3.7 A Sample of cluster from Ontology 1

Constructing Blocks: - In this stage block are constructed by assigning RDF sentences to the clusters. A block can view as a subset of Ontology (i.e. a set of triples). Clusters cannot be directly used for existing Ontology matching approaches because the RDF triples connecting these entities are missing. Therefore we need to recover the triples to the clusters as seen in figure 3.8.

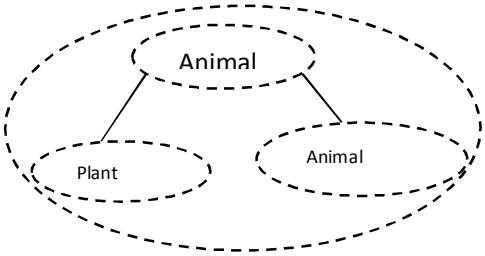


Figure 3.8 Sample of a block from Ontology 1

Using the same process performed on Ontology $_1$, a similar block from is formed in Ontology $_2$.

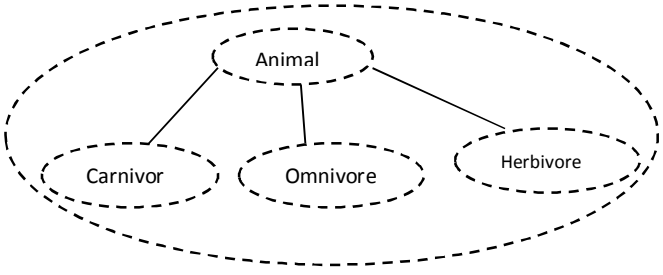


Figure 3.9 Sample of a block from Ontology 2

PHASE 2

Finding Anchors: - Alignments with high similarities are termed anchors. The anchors between large Ontologies are gained using I-sub to automatically generate the set of candidate anchors.

Generating Block Mappings

Similarities between blocks can be computed based on the distribution of anchors gained. The Idea behind is that the more anchors can be found between two blocks the more Similar the two blocks. Matched blocks with high similarities are selected to be further matched by two powerful matchers V-Doc and GMO.

iii. V-DOC (Virtual Document)

The virtual document is considered as a collection of words directly obtained or inferred from the textual content of the document. Virtual document of an entity declared in Ontology contains not only the local descriptions but also the neighboring information to reflect the intended meaning of the entity. Here, a virtual document is generated for each entity declared in OWL/RDF Ontology (Qu, *et al*, 2006).

V-Doc exploits the RDF graph an RDF graph consists of a number of RDF triples. An RDF triple or statement is represented in the form of <subject, predicate, object>.

PROCESS OF VIRTUAL DOCUMENT

Pre-processing: When the Ontologies are inserted an internal representation and validation of the Ontologies is done and saved in before V-DOC matching process. The words involved are normalized. The words involved should be normalized first in a pre-processing step, e.g., parsing literals into tokens through eliminating stop words. (Zhang, *et al*, 2012).

Entity retrieval and classification: In this process all the token nodes and neighbors of each node in each Ontology are found, then categorizes the entities into Classes, Properties and instances and places it within a Classdoclist, Propertydoclist and Instance doclist

Classdoclist = $\{c_1, c_2, c_3 \ldots c_n\}$

Propertydoclist = $\{p_1, p2, p_{3 \ldots pn}\}$

Instancedoclist = $\{I_1 , I_2 \ldots I_n\}$

Descriptions: The description of each entity of the Ontologies using the description formulations is given in (Qu, *et al*, 2006) as:

> **Definition 1:** Let e be a URIref, the description of e is a collection of words defined by (1):
>
> $Des(e) = \alpha 1 \times$ collection of words in the local name of e
>
> $+ \alpha 2 \times$ collection of words in the rdfs:label of e
>
> $+ \alpha 3 \times$ collection of words in the rdfs:comment of e
>
> $+ \alpha 4 \times$ collection of words in other annotations of e *(3.2)*
>
> Where $\alpha 1; \alpha 2; \alpha 3$ and $\alpha 4$ are fixed rational numbers in $[0; 1]$, which indicate the weights for each category. The operator '+' denotes the merging of collections of words on both sides

Neighbors of Entities

To better express the descriptions of the virtual documents, the descriptions of neighbours were grouped. Three neighbouring operations are defined to present different types of neighbours. For an entity *e*, we denote SN (*e*) (subject neighbouring operation) as the nodes placed at the subject position of RDF triples. Similarly, PN (*e*)

(predicate neighbouring operation) and ON (*e*) (object neighbouring operation) are defined as the nodes in the predicate position and object position of triples, respectively.

Virtual Documents:

VD (e) = Des (e)

$$+\gamma_1 * \sum_{e \in SN(e)} Des(e)$$

$$+\gamma_2 * \sum_{e \in PN(e)} Des(e)$$

$$+\gamma_3 * \sum_{e \in ON(e)} Des(e) \qquad (3.3)$$

Where γ_1, γ_2 and γ_3 are rational numbers in the range [0, 1].

The similarities between virtual documents in the two ontologies are computed. If the two virtual document share one or more equal words they are seen as similar, else they are not similar.

Example: Using Ontology $_1$ and Ontology$_2$ and trying to find the similarity between herbivorous and plant eaters.

VD (herbivorous) = Des (herbivorous)

$+ \gamma_1 *$ Des (eats)

$+ \gamma_2 *$ Des (Animal)

VD (herbivorous) = {herbivorous, eats, Plant}

VD (herbivorous) VD (Plant eaters) = Des (Plant eaters)

$+ \gamma_1 *$ Des (eats)

$+ \gamma_2 *$ Des (Animal)

VD (Plant eaters) = {Plant eaters, eats, plant}

Finally, since the virtual document of the herbivorous and plant eaters share similar terms in the derived virtual document, herbivorous and plant eater are similar.

VD (herbivorous) = {herbivorous, eats, plant}

VD (Plant eaters) = {Plant eaters, eats, plant}

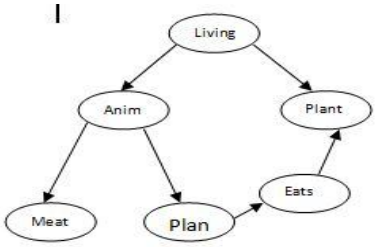
iv. GMO (GRAPH MATCHER FOR ONTOLOGIES)

The Graph matcher for Ontologies is also another important component in Falcon-AO as the name entails it is based on a graph matching approach for Ontologies.GMO uses a directed bipartite graph to represent Ontologies instead of RDF graph model.

The main idea behind GMO is that similarity of two entities from two Ontologies comes from the accumulation of similarities of involved statements (triples) taking the two Ontologies as the same role. I.e. Subject, Predicate, Object in the triple. GMO normally takes a set of matched entity pairs, which are typically found previously by other approaches (V-doc and I-sub) as external mapping input in the matching process and output additional matching pairs by comparing their structural similarities (Hu, *et al*, 2005).

Overview of GMO

From a study by (Jian, *et al*, 2005) this are the step-by step process using a portion of sample Ontology.



i. Parse and, and transform them to corresponding RDF bipartite graphs. *Using a portion of the first Ontology*

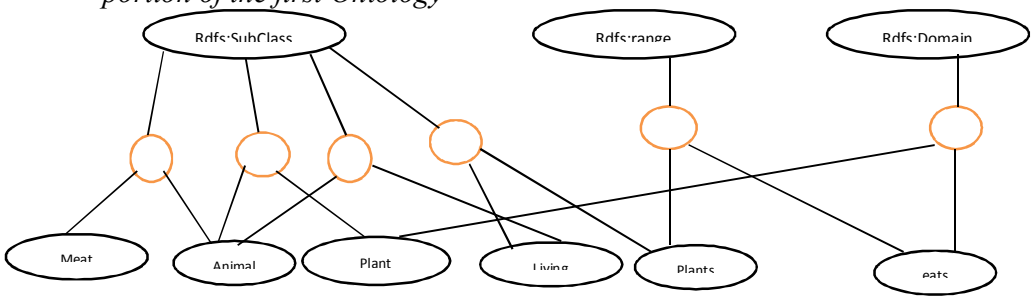


Figure 3.10 RDF bi-partite graph

ii. The directed bi-partite graph is derived by replacing the "s" edges with edges pointing to statement nodes, and the "p" and "o" edges with edges pointing from statement nodes.

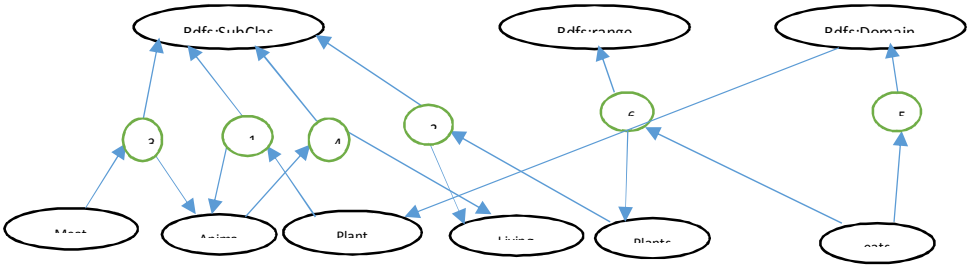


Figure 3.11 Directed bi-partite graph

ii. Classify entities in *OA* and *OB* as classes, properties and Instances

iii. Coordinate using coordination rules.

Due to heterogenic nature in expressing semantic and inference capability of Ontologies, it is necessary to coordinate the two by bringing them into a harmonious relationship before mapping them. Discarding**:** Within Ontology some statement (triples) may become redundant and or useless for structural similarity. Ontology header is not required in structural comparison.

3.4.3 Alignment set

The Alignment set consist of two parts Alignment Generator and Alignment Evaluator. It generates alignments by using a widely accepted format then evaluates the generated alignment against reference alignment (manually generated alignment by domain experts) based on standard precision/ recall metrics.

3.4.4 Central controller

The central controller executes matchers and then merges similarities based on measure of linguistics comparability and structural comparability.

3.4.5 Repository

The Repository is used in storing important information throughout the matching process i.e. Stores reusable data during matching process.

3.4.6 Similarity combination strategy

Similarity Combination is an important and difficult issue in building Ontology matching systems. A strategy to tune the thresholds (or cutoffs) of our matchers gradually based on the measures of linguistic comparability and structural comparability was proposed in this system, which makes the combination rich.

i. **The linguistic comparability**: - is done by measuring the proportion of candidate alignments against the minimum number of entities in two Ontologies. The intuition is as follows:

    1. If the number of alignments is close to the number of entities in the smallest Ontology, we are nearly done with matching, and it is unnecessary to execute GMO anymore.
    2. If the alignments are not enough e.g. for low overlapped Ontologies, the number of alignments might be considerably smaller than the number of entities in the smallest Ontology then it is necessary to execute GMO.

ii. **The structural comparability*:* -** It starts by comparing the built-in vocabularies used in the two ontologies. The basic assumption is the more built-in vocabularies are mutually included in the two ontologies, the more similar they might be in structure. But then measuring this is not enough, it also compare the alignments found by V-Doc or I-Sub with high similarities to the alignments discovered by GMO, thus the reliability of the results of GMO can be estimated roughly.

The two kinds of comparability are divided into three levels (i.e., low, medium and high) to determine the similarity combination strategy automatically.

1. If the linguistic comparability is high, the approach would lower the threshold of V-DOC and I-sub, and then more alignments from V-DOC and I-sub can be combined to the final alignments.

2. If the linguistic comparability is lower than the structural comparability, the outputted alignments largely derived from GMO.
3. If the linguistic comparability and the structural comparability are both low, our approach would combine less alignments from V-DOC and GMO to the final result but make the alignments from V-DOC more than the ones from GMO, because the alignments from V-DOC are assumed to be more reliable than the ones from GMO.
4. Otherwise, the outputted alignments are generated by making a combination among V-Doc, I-Sub and GMO with a weighting scheme.

CHAPTER FOUR

IMPLEMENTATION OF THE EXTENTEDED FALCON-AO

This Chapter explains the implementation phase of the objectives. The implementation done is a modification on an Ontology Alignment System named Falcon-AO. The new component consists of Ontology Parsing, WordNet Database Search, Model creation, Rule Specification and Data Structure Creation of Matched Entities.

4.1 Differences of Falcon-AO and extended Falcon-AO

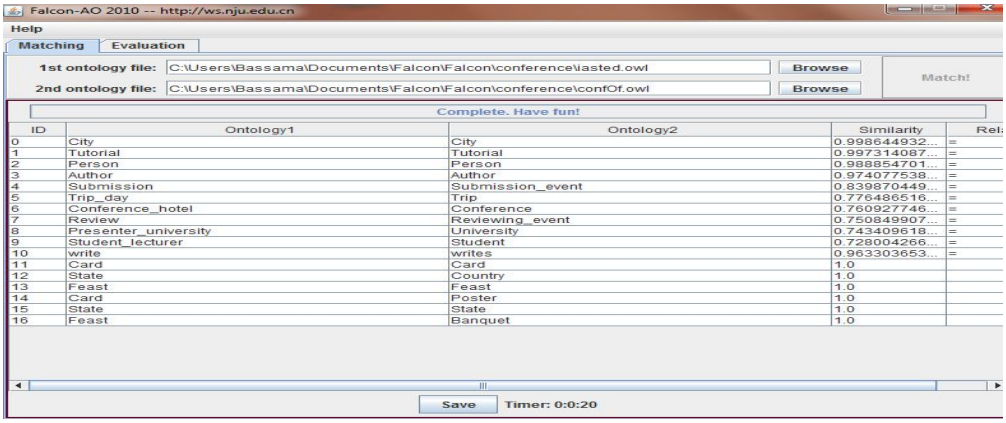


Figure 4.1 Extended Falcon-AO Interface

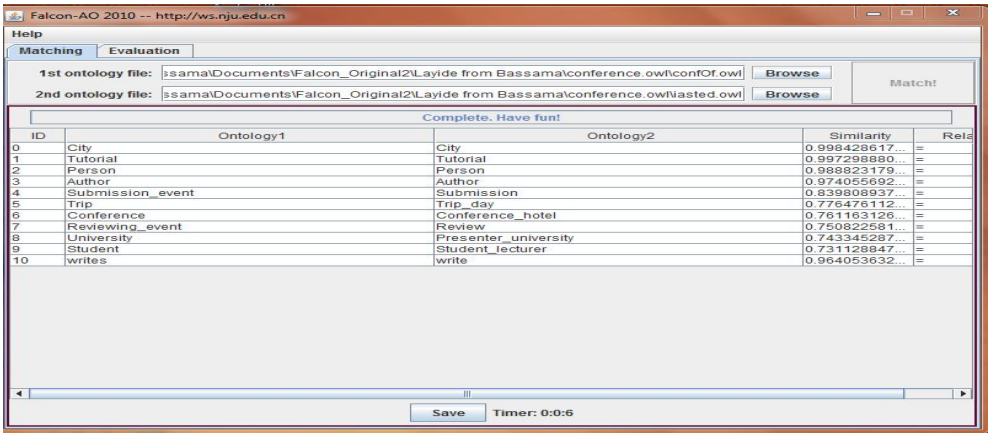


Figure 4.2 Falcon-AO Interface

Figures 4.1 and 4.2 are the snapshots of the Falcon-AO and extended Falcon-AO after Aligning same two ontologies. The two interfaces have been modified to include timing respectively. In the extended Falcon-AO's Graphical user Interface, the time elapsed in

running the task is captured and shown. We shall use the following section to explain the modifications carried out on the original Falcon-AO.

4.2 PROCESSES OF THE EXTENDED SYSTEM

4.2.1 **Timer**

```
            public void actionPerformed(ActionEvent ev)
            {
                Thread th = new Thread()
                {
                    @Override
                    public void run()
                    {
                       ++MatchPanel.timer;
                        int remin=MatchPanel.timer/60;
                        int hr=MatchPanel.timer/3600;
                        clock=hr+":"+remin+":"+(remin < 1? MatchPanel.timer:MatchPanel.timer-(remin*60));

                        MatchPanel.timerLbl.setText("Timer: "+clock);
                    }
                };
                th.start();
            }
    });
    timer_clock.start();
    timerLbl.setBounds(100, 20, 250, 30);

    savePanel.add(timerLbl);


    detailPanel.add(savePanel, BorderLayout.SOUTH);
    //bottomPanel.add(timerLbl);
    Thread match = new Thread()
    {
        @Override
```

Figure 4.3 Timer Code

On clicking the match button, after inserting the two Ontologies, timer functionality is invoked to keep track of the processing time. This timer functionality is threaded in a separate thread from main thread that is carrying out the entity matching. The code snapshot in Figure 4.3 shows how the timer object was created and as well supplied with the time interval in which it is going to update the display on the GUI.

4.2.2 WordNet Processing

```
private void wordNetProcessor()
{
   browseFile1=MatchPanel.textFieldOnto1.getText().trim();
   browseFile2=MatchPanel.textFieldOnto2.getText().trim();
   OntModel mod1=readOntology_2("file:"+browseFile1);
   OntModel mod2=readOntology_2("file:"+browseFile2);

   enumerateClasses(mod1, mod2);


   readOntology("file:"+dest);
```

Figure 4.4 WordNet Processing

In the WordNet Processing method captured in Figure 4.4, the two Ontologies file names are read and kept, subsequently, they are then read and their classes or concepts been retrieved and stored for the matching process, a temporary Ontology file is created to store the entities that WordNet was able to matched, and this we observe on the last line of the code in Figure 4.4, as the temporary Ontology file was also been read. Retrieves all the entities in both Ontologies and then tries to find similarities, the classes retrieved from the first Ontology are stored in a data structure called *keyword1*. For example, if the first Ontology has 40 entities the variable *keywords1* would have 40 entities and same for the second Ontology.

4.2.3 Getting Synonyms from WordNet

```
boolean found=false;
removeClasses();
  for(int n=0; n < keywords1.size(); n++)
      {
           String pick=keywords1.elementAt(n);
           if(pick == null)
               continue;
           ArrayList<String> synonyms_of_pick=getSynonymsFromWordNet(pick);

           for(int k=0; k < keywords2.size(); k++)
                {
                    for(int m=0; m < synonyms_of_pick.size(); m++)
                        {
                            if(keywords2.elementAt(k) == null)
                                continue;
```

```
for(Iterator i=mod2.listClasses(); i.hasNext();)
  {
    ont_class=(OntClass) i.next();
    name=ont_class.getLocalName();
    uri=ont_class.getURI();
    if(uri != null)
      {
        if(ont_class != null && uri.contains(browseFile2.substring(index, next_index)) ) //&& classname3.getURI().contains(dest.substring(index)))
          {
             keywords2.add(name);
             System.out.println("2 == "+name+" ==");
          }
      }
    if( k > 50)
       break;
    ++k;
  }
```

Figure 4.5 Getting Synonyms from WordNet

We have retrieved all the classes in the two Ontologies from Figure 4.4 and stored the classes of the first Ontology in *keywords1* and that of the second Ontology in *keywords2*. Now, Figure 4.5 iteratively walks through the entities of the first Ontology and then finds the synonyms of each of those entities. For each of the synonyms of each of the entities in the first Ontology, a corresponding matched entity is search for from

the second Ontology. Whenever there is a match, the two entities from the two Ontologies marked as satisfying the requirement for matching.

4.2.4 How Synonyms Are Retrieved From WordNet

```
private ArrayList<String> getSynonymsFromWordNet(String word)
  {
    //word="good";
    //System.out.println("ffffffff "+word);
    NounSynset nounSynset;
    NounSynset[] hyponyms;

    ArrayList<String> syn=new ArrayList<String>();
    WordNetDatabase database = WordNetDatabase.getFileInstance();
    Synset[] synsets = database.getSynsets(word, SynsetType.NOUN);
    if (synsets.length > 0)
    {
      for (int i = 0; i < synsets.length; i++)
        {
          nounSynset = (NounSynset)(synsets[i]);
          hyponyms = nounSynset.getHyponyms();

          String[] wordForms =nounSynset.getWordForms();
          for (int j = 0; j < wordForms.length; j++)
```

```
      String[] wordForms =nounSynset.getWordForms();
      for (int j = 0; j < wordForms.length; j++)
        { //System.out.println(wordForms[j]+" "+synsets.length);

          // System.err.println(nounSynset+" =>  "+nounSynset.getWordForms()[j] +
            //     ": " + nounSynset.getDefinition() + ") has " + hyponyms.length + " hyponyms");

         if(!syn.contains(nounSynset.getWordForms()[j]) || !word.equalsIgnoreCase(nounSynset.getWordForms()[j]))
             syn.add(nounSynset.getWordForms()[j]);
        }

      }
    syn.add(word);
  }
  return syn;
}
```

Figure 4.6 How Synonyms are retrieved from WordNet

WordNet Database is a lexicon which provides access to the WordNet database information. It includes methods that can access the WordNet database by searching files on the local file system. Returns all synsets, which contains the specified word form. In this case the synsets should be word and the type should be noun. It retrieves the word forms and array of synonymous words. In the implementation process, a method was created to actualize this task. This method is called from another method named wordNetProcessor. The WordNet processor retrieves all the entities in $Ontology_1$ and $Ontology_2$, it picks all the entities in Ontology then sends it to the getSynonym method, so as to retrieve all the synonyms of the class passed to it. It then saves all the synonyms as an arraylist of String.

*4.2.*5 Implementing Same AS rule

```
        //if(type != 2){
            OntClass classB=m1.createClass(prefix1+entity2);    //prefix2+
        //}
        classA.addSameAs(classB);
        //generateRules(entity1, entity2, constraint, pos);

      }
     else
      {
        //JOptionPane.showMessageDialog(null,  "This two entities exist already", "Error", JOptionPane.ERROR_MESSAGE);
      }

  }
```

Figure 4.7 Implementing SameAs rule

Once two entities from the two different Ontologies have been found to match by the WordNet, they are then temporarily stored in the temporary Ontology using the sameAs rule provided by Jena. This was achieved by the code snippet shown by Figure 4.7, where the addSameAs method was used to indicate that two given entities should be modeled as equivalent in the temporary Ontology.

4.2.6 Getting Equivalent Entities

```
private void getEquivalentEntities()//Property prop
{  //getDerivation(null)
    model1=m1.read(file1);
   for(Iterator i=m1.listClasses(); i.hasNext();)
    {
       OntClass classname=(OntClass) i.next();
       String name=classname.getLocalName();

       if(constraint_pairs2.contains(name) || constraint_pairs1.contains(name))
            continue;

       OntResource classname3=classname.getSameAs(); //.getEquivalentClass(); //

       if(classname3 != null && classname != null && classname3.getURI() !=null && classname.getURI() !=null) //&& classname3.hasLiteral(prop, "co
       {
            int index=dest.lastIndexOf("/")+1;
            int next_index=dest.lastIndexOf(".");
            if(classname.getURI().contains(dest.substring(index, next_index)) && classname3.getURI().contains(dest.substring(index, next_index)))
            {
              System.out.println(classname3.getLocalName()+" &&&& "+classname.getLocalName());
              String name2=classname3.asNode().getLocalName();
             if( !constraint_pairs1.contains(name)  &&  !constraint_pairs2.contains(name2))
                {
                    String nameA=classname.asNode().getLocalName();
                    name2=classname3.asNode().getLocalName();

                    OntModel modelA=(OntModel)classname.getModel();
                    RBGModel rbgm1 = RBGModelFactory.createModel("");
                    rbgm1.setOntModel(modelA);

                    OntModel modelB=(OntModel)classname3.getModel();
                    RBGModel rbgm2 = RBGModelFactory.createModel("");
                    rbgm2.setOntModel(modelB);

                  constraint_pairs1.add((Node)rbgm1.getNode(classname3));
                  constraint_pairs2.add((Node)rbgm2.getNode(classname));
                }
       }
  }
```

Figure 4.8 Getting Equivalent Entities

It is expected that WordNet will loop through all the entities in the first Ontology and then systematically attempt to find the entities of the second Ontology that are equal to the synonyms of the entities of this first Ontology . After all these process have been carried out and the sameAs specified for similar entities from both Ontologies, these

similar entities that have been modeled in the temporary Ontology are then retrieved as paired equivalent classes which are then stored in a data structure for merging with what Falcon-AO have been able to match. Figure 4.8 shows the snapshot of code implementation that was used to achieve this. We see how the getSameAs() Java method was used to pick out the equivalent class of each corresponding class chosen from the temporary Ontology . The variables or data structure that holds all the matched entities are called constraint_pair1 and constraint_pair2.

4.2.7 How Matched Entities Are Mapped

```
align=new Alignment(); //import nju.websoft.falcon.model.Node;
System.out.println("List of matches from Wordnet. Size is= "+wordnet_constraints.constraint_pairs1.size());
for(int i=0; i < wordnet_constraints.constraint_pairs1.size(); i++)
{
       Node node1=wordnet_constraints.constraint_pairs1.get(i);
       Node node2=wordnet_constraints.constraint_pairs2.get(i);
       align.addMapping(new Mapping(node1, node2, 1.00,""));
       System.out.println(node1.getLocalName()+" == "+node2.getLocalName());

}
addUserSpecifiedConstraint();
```

Figure 4.9 How Matched entities are mapped

Entities matched by Falcon-AO are stored in a Java data structure called *Mapping*. We tried to subsequently merge entities matched by WordNet with those once matched by Falcon-AO. Therefore, we prepare the WordNet matched entities to take the format of the Java *Mapping* data structure so that we may be able to easily provide users with the overall matched entity from the extended Falcon -AO.

Note that all that Falcon-AO has matched are been saved in a variable called *alignment* which is an instance of a class called Alignment. We also created a variable called a*lign* which is an instance of Alignment so that at the end of the process what Falcon-AO has matched and what WordNet has matched would be merged together.

4.2.8 Merging WordNet and Falcon-AO Matched Entities

```
    private void addUserSpecifiedConstraint()
    {
        int k=0;
        while(k < MatchPanel.align.size())
        {         //JOptionPane.showMessageDialog( null, "Inside user constraint adder!" );
            alignment.addMapping(align.getMapping(k));
            ++k;
        }
    }
}
```

Figure 4.10 Merging WordNet and Falcon-AO matched entities

All entities Matched by WordNet are here joined with entities that Falcon-AO has been able to match. The eventual merged entities would be displayed as the result of embedding WordNet in Falcon-AO for the purpose of improving the semantic relationship of the concepts in the two Ontologies. The variable named *alignment* in figure 4.10 contains the entities Matched by Falcon-AO while the variable named *align* contains the entities matched by WordNet.

4.3RESULTS AND DISCUSSION

4.3.1OAEI (Ontology Alignment Evaluation Initiatives)

The increasing number of methods available for Ontology alignment mandates an agreement for evaluation of these methods (Euzenat, 2013). It evaluates the alignment techniques based on standard matrices precision, recall and F-Measure, but in the cause of this thesis we are going to use only the precision. The OAEI also provides Standard benchmark Ontology data sets for tests.

4.4.2 Experimental setup

Hardware

The tests were performed on an Intel(R) Celeron(R) 2.20 GHz Processor ,3.00 GB of RAM and 250 GB Hard Disk.

Software

The system has windows 7 Home Premium, the application was built on Netbeans IDE, WordNet as the lexical database and Java API for WordNet and Jena API.

4.4 EVALUATION CRITERIA

We considered matching of classes and instances based on their similarity in terms of meaning in the two input RDF(s) or OWL-ontologies. As match quality measures, we used the following indicators: Precision and Time.

4.4.1 Precision

Is a value in the range [0, 1]; the higher the value, the fewer the wrong mappings computed.

The precision measure could also be defined as follows:

$$\textbf{Precision} = \frac{number_of_correct_found_Alignmnts(bytools)}{number_of_found_Alignmnts(bytools)} \qquad (4.1)$$

4.4.2 Time

Is the value of time taken for the matching process after inputting the two Ontologies.

For the running time, we conducted numerous experiments to show the impact of the proposed framework on the overall performance and no other application running but the proposed system.

Table 4.1 Results of the Experiment

Input			Without WordNet				With WordNet			
S/N	Ontology 1	Ontology 2	Time in seconds	Precision	Found	Correct	Found	Correct	Precision	Time in Seconds
1	Cmt	Conference	4	0.5	18	9	19	10	0.5263158	8
2	Confof	Ekaw	5	0.61904764	21	13	23	15	0.65217394	9
3	Confof	Iasted	5	0.36363637	11	4	14	7	0.5	9
4	Ekaw	Iasted	6	0.53846157	13	7	14	8	0.5714286	9
5	Edas	Iasted	8	0. 6363636	11	7	13	8	0.61538464	14
6	Cmt	Confof	4	0.54545456	11	6	11	6	0.54545456	7

4.5 DISCUSSION OF RESULTS

The Table 4.1 depicts the results of matching Ontologies included in the experiment. The Ontologies are from the Ontology Alignment Evaluation Initiative from the conference Ontologies and using one of the standard measures of OAEI. The Table shows that six different pairs of Ontologies were used and Alignment with WordNet alongside syntactic string-based similarity measures does improve the quality of the alignment , it demonstrated similar or better precision, it also showed an increase in execution time, the first five pairs all show an increase in time and precision in aligning the Ontologies but the last pair showed an increase in time but no increase in precision this indicates that even when no new match entities are found by the system the time increases due to additional computation, So there is a tradeoff between time and precision.

4.6 SUMMARY

The chapter started by introducing the graphical user interface of the two systems and then proceeded to explain the processes in achieving the extended system, then the results were discussed using the time and the precision as the indicators and then finally the conclusion was given indicating an increase in processing time and precision.

CHAPTER 5

SUMMARY, CONCLUSION AND RECOMMENDATION

This Chapter discusses the conclusion of what has been done within this thesis. The Section summarizes the contribution of this thesis with theoretical, experimental and implementation contributions. It gives an overview regarding goals and their accomplishment. The thesis is wrapped up with remarks about future research.

5.1 SUMMARY AND CONCLUSION

There is need to make the Web better and this can be done by adding semantics to it through Ontologies but these Ontologies have existing problems of increasing number and heterogeneity. This problem can only be solved using a process called Ontology Alignment and is done by a number systems which use many different approaches but does not always satisfy the problem of the Ontologies especially when the structure of the Ontologies are different or when semantics is required.

The main objective of this work is to modify an Ontology Alignment System named Falcon-AO with WordNet which is a lexical knowledge source to enhance the matching result of the system. The two Ontologies to be matched are input into the system and then passes through the WordNet where synonymous terms in the two Ontologies are detected and then passes through the matcher library where the other four existing Matchers are to find Aligning terms (entities) in the two different Ontologies.

In the Extended Falcon-AO the user starts by Inserting two different Ontologies of the same domain the Ontologies are then read and all entities of the Ontologies are retrieved and saved in different storage .when the entities are retrieve those of the first Ontology are picked one by one and passed to WordNet to retrieve their synonyms, for each entity whose synonyms have been retrieved the synonyms are searched to check if there is a

term similar to those entities in the second Ontology, If matched terms are found they are store in a different container where rules are to be applied to them later. Those entities with similar meaning are seen as equivalent entities and the rule is applied to each pair of synonymous terms. When these entities are matched they have to be casted to a specific data structure so that those entities matched by WordNet and the other matchers can be combined

Secondly the extended Falcon-AO system was evaluated beside the Falcon-AO based on precision and time using benchmark Ontologies from the Ontology Alignment Evaluation Initiative. Though, there was increase in time due to additional computation but the results were satisfying due to increase in precision.

From the description we provided on our extended system, it is clear that the system achieved all the stated objectives and answered the questions of the research work.

i. Can WordNet be integrated into Falcon-AO?

   Yes, WordNet can be integrated to Falcon-AO and can also provide appreciating results.

ii. Can Falcon-AO's precision be enhanced using WordNet with acceptable execution time penalty?

   The systems precision was increased but there a considerable increase in time due to increase in computation.

5.2 LIMITATIONS AND FUTURE WORK

This section highlights some possible future directions for advancing the Ontology Matching System we have extended.

i. The system deals only with synonymous terms in future WordNet should be exploited to detect initially falsely detected correspondences, e.g. by taking antonyms. Although this step will not add semantics to the mapping, it is potentially able to increase its precision.

ii. The tuning of the algorithms within Falcon-AO is still a rigid process. For example, PBM performs well on the large Ontologies with simple class hierarchy structures. But when the relations in Ontologies are complicated the partitioning quality of PBM is not sound.

iii. Falcon-AO can be Improved to support not only 1:1 mapping

REFERENCES

Cao, Y., Wang, X., Zhang, F., & Yang, W. (2012). Ontology-based Domain Knowledge Acquisition Technology. *2012 Fifth International Symposium on Computational Intelligence and Design*, (0) 2, 487-490.

Clerkin, P., Cunningham, P., & Hayes, C. (2002) *Ontology Discovery for Semantic web Using Heirarchical Clustering.* Semantic Web Mining Workshop, 1-8.

Ehrig, M., & Staab, S. (2004). *QOM:Quick Ontology Mapping.* Hiroshima: Third International Semantic Web Conference, 356-361.

Euzenat, J. (2013, October 3). *Ontology Alignment Evaluation Initiative*. Retrieved December 10, 2013 from http://oaei.ontologymatching.org/

Euzenat, J. Shvaiko, P., & J´erome. (2013). Ontology Matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering, 25(1)* 1-20.

Gruber, R.T. (1993). *A Translation Approach to Portable Ontology.* Knowledge Systems Laboratory Computer Science Department, Stanford University, California.

Hamdi, F., Safar, B., & Niraula, N. B. (2009). TaxoMap in the OAEI 2009 alignment contest. *Proceedings of the ISWC'09 Workshop on Ontology Matching* , 1-8.

Hassan, F. M., Ghani, I., Faheem, M., & Hajji, A. A. (2012). Ontology Matching Approaches for eRecruitment. *International Journal of Computer Applications*, 51(2) 40-45.

Hu, W. (2010, September 8). *Falcon-AO*. Retrieved July 10, 2013, from Falcon-AO: ws.nju.edu.cn/Falcon-ao/.

Hu, W., Jian, N., Qu, Y., & Wang, Y. (2005). GMO: A Graph Matching for Ontologies. *Proceedings of K-CAP Workshop on Integrating Ontologies*, 41–48.

Hu, W., Qu, Y., & Gong, C. (2008). Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering*, (67) 140- 160.

Hussain, M. M., & Srivatsa, S. (2012).A study of Different Ontology matching system. *International Journal of Computer Applications*, 37(8), 10-16

Jain, V., & Dr. Singh, M. (2013). Ontology Based Information Retrieval in Semantic Web: A Survey. *International Journal of Information Technology and Computer Science*. (10) 62-69.

Jean-Mary, Y., & Kabuka, M. R. (2009). Ontology matching with semantic verification. *Journal of Web Semantics*, 235-251.

Jian, N., Hu, W., Cheng, G., & Qu, Y. (2005). Falcon-AO: Aligning ontologies with Falcon. *Prceedings of K-CAP workshop on integrating Ontologies*, 85-91.

Kaushal, G. (2011). Role of Ontology in Semantic Web. *Journal of Library and information Technology,* 116-120.

Klyne, G., & Carroll, J. J. (2003, January 23). *Resource Description Framework (RDF):Concepts and Abstract Syntax*. Retrieved July 12, 2013, from http://www.w3.org/TR/2003/WD-rdf-concepts-20030123

Lin, F., & Sandkuhl, K. (2008). A Survey of Exploiting WordNet in Ontology Matching. *The International federation for information Processing*, 341-350.

Marta, S., d'Aquin, M., & Enrico, M. (2006). Using the semantic web as background knowledge for Ontology Mapping. *1st International Workshop on Ontology Matching*, 3-4.

McGuinness, Debora. L., & Harmelen, F. (2004, February 10). *OWL Web Ontology Language Overview*. Retrieved August 12, 2013, from http://www.w3.org/TR/2004/REC-owl-features-20040210/

Hanif, M.D., & Masaki, A. (2009). Alignment Results of Anchor-Flood Algorithm. *Proceedings of Ontology Matching Workshop of the 8th International Semantic Web Conference, Chantilly, VA, USA*, 1-8

Moa, M. (2008). *Ontology Mapping :Towards Semantic Interoperability in Distributed and Heterogeneous Environments*. PhD thesis, University of Pittsburgh, 1-147.

Noy, N. F., & McGuinness, D. (2001). *A Guide to Creating Your First Ontology*. Standford Knowledge Systems Laboratory Technical Report KSL-01-05 And Stanford medical informatics Technical Report ,Stanford University.

Qu, Y., Hu, W., & Cheng, G. (2006). Constructing Virtual Documents for Ontology Matching. *Proceedings of the 15th International World Wide Web Conference*, 23-31.

Saruladha, K., Aghila, G., & Sathiya, B. (2011). A Comparative Analysis of Ontology and Schema Matching Systems. *International Journal of Computer Applications*. 34(8), 14-21.

T. Berners-Lee, M. F. (1999). Weaving the Web: The original Design and ultimate Destiny of the World Wide Web.

Tang, J., & Liang, B. (2005). Towards Detecting Mapping Strategies For Ontology Interoperability, 1-8.

Taye, M. M. (2010). Web-Based Ontology Languages and its Based Description Logics. The Research Bulletin of Jordan ACM, 2(2), 11-19.

Wang, Y., Liu, W., & Bell, A. (2010). A Structure-based Similarity Spreading Approach for Ontology Matching, 361-374.

Zhang, H., Wei, H., & QU, Y. (2012). V-DOC+: a virtual document based approach for matching large Ontologies using MapReduce. *Journal of Zhejiang University-Science (Computers & Electronics)*.13(4), 257-267.

APPENDIX A

Retrieving Synonyms from WordNet

```
private ArrayList<String> getSynonymsFromWordNet(String word)
   {
    //word="good";
    //System.out.println("ffffffff "+word);
    NounSynset nounSynset;
    NounSynset[] hyponyms;
    ArrayList<String> syn=new ArrayList<String>();
    WordNetDatabase database = WordNetDatabase.getFileInstance();
    Synset[] synsets = database.getSynsets(word, SynsetType.NOUN);
    if (synsets.length > 0)
    {
      for (int i = 0; i < synsets.length; i++)
        {
          nounSynset = (NounSynset)(synsets[i]);
          hyponyms = nounSynset.getHyponyms();
          String[] wordForms =nounSynset.getWordForms();
          for (int j = 0; j < wordForms.length; j++)
            {
hyponyms");


              if(!syn.contains(nounSynset.getWordForms()[j]) ||
!word.equalsIgnoreCase(nounSynset.getWordForms()[j]))
                  syn.add(nounSynset.getWordForms()[j]);
            }
             }
       syn.add(word);
   }
private void wordNetProcessor()
   {
```

```
browseFile1=MatchPanel.textFieldOnto1.getText().trim();

browseFile2=MatchPanel.textFieldOnto2.getText().trim();

OntModel mod1=readOntology _2("file:"+browseFile1);

OntModel mod2=readOntology _2("file:"+browseFile2);

String pos1=browseFile1.substring(browseFile1.lastIndexOf("\\")+1,
browseFile1.lastIndexOf(".owl"));

String pos2=browseFile2.substring(browseFile2.lastIndexOf("\\")+1,
browseFile2.lastIndexOf(".owl"));

System.out.println(pos1);

System.out.println(pos2);

prefix1="http://"+pos1+"#";

prefix2="http://"+pos2+"#";

enumerateClasses(mod1, mod2);

 readOntology ("file:"+dest);

 boolean found=false;

removeClasses();

                for(int n=0; n < keywords1.size(); n++)

                {

                   String pick=keywords1.elementAt(n);

                   if(pick == null)

              continue;

          ArrayList<String> synonyms_of_pick=getSynonymsFromWordNet(pick);


          for(int k=0; k < keywords2.size(); k++)

             {

               for(int m=0; m < synonyms_of_pick.size(); m++)

                 {

                   if(keywords2.elementAt(k) == null)

                         continue;


                   if(synonyms_of_pick.get(m).equalsIgnoreCase(keywords2.elementAt(k)) &&

                        !(pick.equalsIgnoreCase(keywords2.elementAt(k))))
```

```
                    {
                      addUserConstraints(pick, keywords2.elementAt(k), "Equivalent",
k+"_"+m);
                      found=true;
                      //break;
                    }


                  }
                }
              }
            //applyRules(all_rules.toString(),(OntModel)m1);
```